Bioinformatics tools for the visualization and structural analysis of metabolic networks

Von der Fakultät für Lebenswissenschaften

der Technischen Universität Carolo-Wilhelmina

zu Braunschweig

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

genehmigte

D i s s e r t a t i o n

von Márcio Rosa da Silva

aus Porto Alegre, Brasilien

1. Referent: Prof. Dr. Wolf-Dieter Deckwer
2. Referent: Prof. Dr. An-Ping Zeng
eingereicht am: 3.5.2006
mündliche Prüfung (Disputation) am: 6.7.2006

2006

(Druckjahr)

Acknowledgments

I would like to express my gratitude to my supervisor Prof. Dr. An-Ping Zeng for his guidance and constructive criticism during all this work.

I am grateful to Prof. Dr. Wolf-Dieter Deckwer for being my mentor and also acting as referee for this work.

I also would like to thank Prof. Dr. Siegmund Lang for acting as the third Prüfer for this work and also for his advice during the process for the submission of this work.

Thanks to all my colleagues in the Systems Biology group in the GBF: to Ahmed Haddad, Bharani Kumar, Feng He, Ping Zheng, and in especial to Dr. Hongwu Ma and Dr. Jibin Sun for the fruitful discussions for improvement of this work.

Thanks to Angela Walter for all her help, especially with “German burocracy”.

To all Brazilian friends that lived here in Braunschweig: Irineu, Linda, Giordano, Sofia, Adriano, Debora, Alexandre, Silvana and all others for making life a little easier in a foreign country with their friendship. Thank you!

I also would like to thank all friends we made here: Evelyn, Birgit, Matthias, Coni and all others who teach us a little of the German culture.

I’d like to thank Unisinos and CAPES for the financial support for this work.

Thanks to my family, my father Volnei, my mother Enedina and my sister Patricia for always support me.

Last but not least, I would like to thank my daughter Isabela for always making me smile even in the most difficult situations and for my wife Rosângela who always believed in me. Thank you two for your love! I love you!

Thank you all very much!

Table of contents

List of figures

List of tables

List of listings

Chapter 1

Introduction

Living organisms are hierarchical structures that integrate their smallest constituent parts – individual molecules including DNA, proteins and metabolites – across multiple levels of organization, from organelles to cells, tissues, organs, and the organism. Thus, a major challenge in biology and medicine today is to understand how the large numbers of different molecular parts interact and self-organize into a whole system that exhibits organic properties that cannot be explained solely in terms of their component properties.

Systems Biology is an academic field that seeks to integrate different levels of information to quantitatively understand how biological systems function. Bioinformatics can help this task by developing tools to analyze and visualize these systems.

In this work, several bioinformatics tools were developed to help the visualization and analysis of biologic networks from a Systems Biology point of view. Methods for the decomposition of metabolic networks are also proposed, including decomposition into a core-periphery structure.

The tools developed include: tools to decompose networks into smaller, functionally related, parts (modules); tools to help visualizing large amounts of data generated by network reconstruction based on genomic and functional genomic data. Software to help the integration of existing tools such as translators for the different file formats used was also developed.

Basic analyses of metabolic networks were also provided as tests for the efficiency of the tools developed and to show how the use of these tools can help network analysis.

In this chapter, an introduction to the terms and concepts used during the work is given to help the understanding of the other parts of the work. As the work is aimed at a Systems Biology approach for metabolic network analysis, an overview of these two topics is first given as well an introduction to graph theory concepts used in this work.

1.1 Systems Biology

It is often said that biological systems, such as cells, are *complex systems*. A popular notion of complex systems is of very large numbers of simple and identical elements interacting to produce *complex* behaviors. Systems biology is the study of an organism, viewed as an integrated and interacting network of genes, proteins and biochemical reactions which give rise to life. Instead of analyzing individual components or aspects of the organism, such as sugar metabolism or a cell nucleus, systems biologists focus on all the components and the interactions among them, all as part of one system (Kitano 2002a). These interactions are ultimately responsible for an organism's form and functions. For example, the immune system is not the result of a single mechanism or gene. Rather the interactions of numerous genes, proteins, mechanisms and the organism's external environment, produce immune responses to fight infections and diseases. Traditional biology has focused on identifying individual genes, proteins and cells, and studying their specific functions. But that kind of biology can yield relatively limited insights into complex systems like the human body.

As an analogy, if one wants to study a car, and focus on identifying the engine, gears, lights and wheels, and study their specific functions, he would have no real understanding

of how a car operates. More important, he would have no understanding of how to effectively service the vehicle when something malfunctions. So too, a traditional approach to studying biology and human health has left us with a limited understanding of how the human body operates, and how we can best predict, prevent, or remedy potential health problems (Battail 2005). Biologists, geneticists, and doctors have had limited success in curing complex diseases such as cancer, HIV, and diabetes because traditional biology generally looks at only a few aspects of an organism at a time.

The individual function and collective interaction of genes, proteins and other components in an organism are often characterized together as an interaction network. Indeed, understanding this interplay of an organism's genome and environmental influences from outside the organism is crucial to develop a (systems) understanding of an organism that will ultimately transform our understanding of human health and disease.

So, the first step towards a systems-level understanding of biology is to move away from a bottom-up to a top-down approach (Junker & Schreiber 2006). Bottom-up means to look at one element of the system (gene, protein/enzyme, metabolite, etc.) and to find out the connections to the neighbors, roles in all processes that the element is involved in, and mechanisms of action.

In contrast, top-down means to first make a snapshot of all elements at a certain level (genes, transcripts, and/or proteins). For this task, since the 1990s many massively parallel experimental techniques have been developed.

Systems biology begins with the study of genes and proteins in an organism using high-throughput techniques to quantify changes in the genome and proteome in response to a given perturbation (Kitano 2002b). High-throughput techniques to study the genome include microarrays to measure the changes in mRNAs. High-throughput proteomics methods include mass spectrometry, which is used to identify proteins, detect protein modifications, and quantify protein levels.

In contrast to much of molecular biology, systems biology does not seek to break down a system into all of its parts and study one part of the process at a time, with the hope of being able to reassemble all the parts into a whole. Using knowledge from molecular biology, the systems biologist can propose hypotheses that explain a system's behavior (2006b). Importantly, these hypotheses can be used to mathematically model the system. Models are used to predict how different changes in the system's environment affect the system and can be iteratively tested for their validity. New approaches are being developed by quantitative scientists, such as computational biologists, statisticians, mathematicians, computer scientists, engineers, and physicists, to improve our ability to make these high-throughput measurements and create, refine, and retest the models until the predicted behavior accurately reflects the phenotype seen.

1.2 Metabolic network

A metabolic network consists of metabolites that are converted into each other by biochemical reactions catalyzed by enzymes. Thus, in a general approach of visualization and analysis, it can be represented as a graph with two different kinds of vertices in alternating order, which is typical for bipartite graphs (see Section 1.3.13.2). Metabolic networks of several microorganisms have been determined through biochemical experiments over the last few decades, and they can be found in various kinds of biochemistry textbooks. For simplicity and for a more straightforward analysis, metabolic networks can be represented as a simple directed graph with nodes representing metabolites (Jeong *et al.* 2000; Ma & Zeng 2003). As a complement, in this case, a reaction graph can also be used (Ma *et al.* 2004). In the reaction graph, the nodes represent reactions and there will be a link between two nodes if the products of the first reaction are substrates in the second.

This simplification helps by opening more opportunities of analysis as many graph algorithms do not consider different types of nodes as in a bipartite graph. The drawback of this representation for metabolic networks is that if the currency metabolites (Ma & Zeng 2003) are not removed, this simplification may create inexistent shortcuts in the network as in the case shown in Figure 1.1 where there is a “shortcut” from glucose to

pyruvate through ADP. Looking only to the network one could interpret that there are only 2 steps to convert from *glucose* into *pyruvate* instead of 9 (Ma & Zeng 2003). So a tool to automatically deal with such networks may present wrong results. By the elimination of the currency metabolites this problem is minimized.

(a) (b)

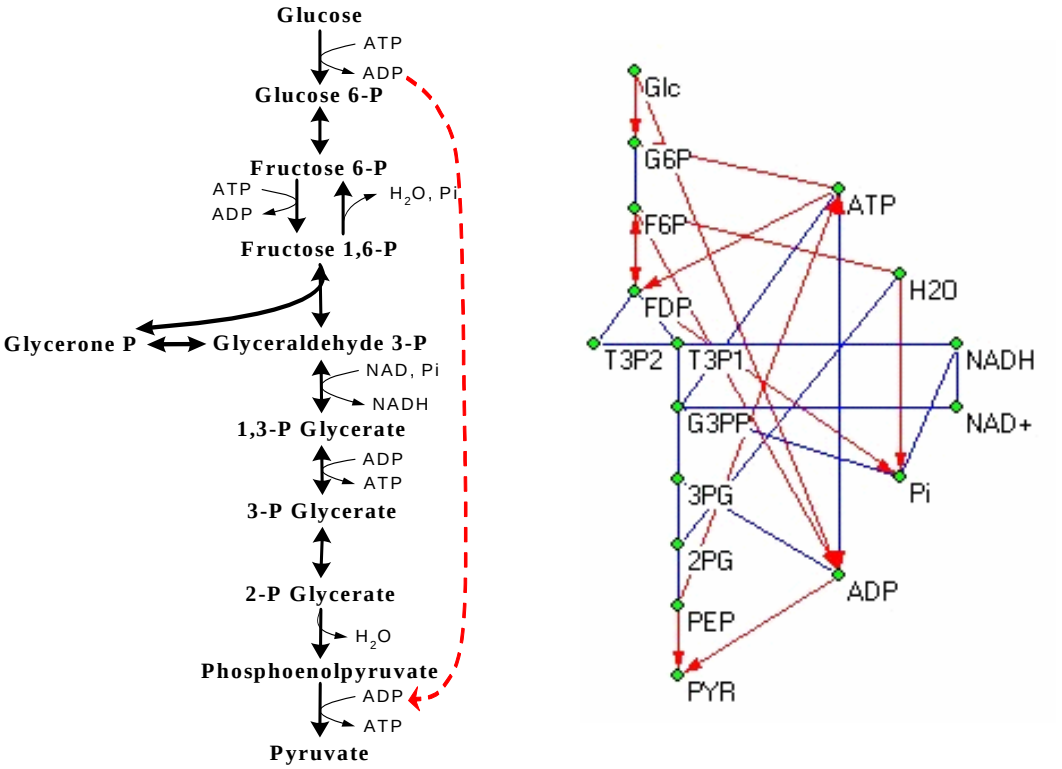


Figure 1.1 – The glycolysis pathway as a part of metabolic network.
(a) The conventional way of presentation in biochemistry; (b) the connection structure in a graphic representation with current metabolites also as connections. (Ma & Zeng 2003)

A summary of biochemical pathways is given in the well-known Boehringer metabolic pathway map (2006a). Since a few years, metabolic pathways have also been predicted from the genome of fully-sequenced organisms. The KEGG database (Kanehisa & Goto 2000) is a public resource for these predicted pathways. In an early study it was elucidated that the structure of the core metabolic network from 43 organisms is identical, being dominated by the same highly connected substrates (Jeong *et al.* 2000) — what was found later (Ma & Zeng 2003) as being the currency metabolites. For the same set of

metabolic networks, it has been stated later that they are organized into many small, highly connected topologic modules that combine in a hierarchical manner into larger, less cohesive units (Ravasz *et al.* 2002).

Several other studies have compared the structure of the metabolic networks of several organisms in order to derive information about their phylogenetic relationship (Podani *et al.* 2001; Forst & Schulten 2001; Ma & Zeng 2004). While these first studies could not replicate the detail of phylogenetic studies based on sequence information, it was at least possible to deduce from the network whether an organism belongs to the domains of Archaea, Bacteria or Eukaryotes.

Efforts have been made to analyze the structure of whole metabolic networks. For example, Schilling *et al.* and Edwards *et al.* (Schilling *et al.* 1999) and (Edwards *et al.* 2002) proposed to use flux balance analysis for predicting the flux distribution in a whole metabolic network under certain physiological conditions. However, this method gives little information about the network overall structure (Ma & Zeng 2003).

Several methods such as elementary flux mode analysis and extreme pathway analysis (Schilling *et al.* 2000; Schilling & Palsson 2000; Schuster *et al.* 1999; Schuster *et al.* 2000) have been developed for analyzing the pathway structure of metabolic networks. These methods have been shown to be useful tools for investigating the metabolic capacity and pathway structure of the metabolic networks (Schuster *et al.* 2000; Papin *et al.* 2002; Stelling *et al.* 2002; Palsson *et al.* 2003; Price *et al.* 2003). These methods are, however, hampered by the combinatorial explosion problem when applied to large-scale networks such as those reconstructed from genomic data. For large-scale networks reconstructed from genome information, decomposition methods should be first used to divide the whole network into small subsystems. The pathway structure of these subsystems may then be properly analyzed by these methods (Schilling & Palsson 2000; Schuster *et al.* 2002).

Methods based on graph theory were shown to be useful for network structure analysis (Jeong *et al.* 2000; Albert *et al.* 2000). In these methods the metabolites correspond to nodes in the graph, and reactions correspond to connections between these nodes. Using

such a graphic representation, Barabasi and his coworkers have studied the structure of metabolic networks using methods adopted from studies of the world wide web (Jeong *et al.* 2000). They found that like all the other networks studied, metabolic networks exhibited typical characteristics of small world networks. This means that most of the nodes have a low connection degree, while few nodes have a very high connection degree. The connection degree distribution follows a power law. The high degree nodes dominate the network structure and are called hubs of the network. Most of the nodes are connected through them by a relatively short path (Strogatz 2001).

For metabolic network, Jeaong et al (Jeong *et al.* 2000) calculated the average path length (AL), which is defined as the shortest path length averaged for every pair of metabolites in the whole network. For 43 organisms they found that AL is almost the same (about 3.2) for all the organisms. This means that most of the metabolites can be converted to each other in about only 3 steps. These results are surprising and in fact unexpected in view of the often long pathways for the synthesis of many metabolites.

Ma and Zeng (Ma & Zeng 2003) showed that by eliminating the currency metabolites, the AL value is different for many organisms and the elimination of these metabolites resulted in a more biologically meaningful network avoiding problems like the shortcuts mentioned earlier in this section (see Figure 1.1).

1.2.1 Modularity of metabolic network

In biochemistry, it is well established that modules consisting of several interacting bioreactions or metabolic pathways build discrete functional units of metabolism (Neidhardt *et al.* 1990; Hartwell *et al.* 1999). These modules are further nested to form a complex metabolic network. Metabolic networks present a small-world structure (Jeong *et al.* 2000; Ma & Zeng 2003) but have also a high clustering coefficient (Ravasz *et al.* 2002). To resolve the apparent contradiction between the small-world structure and modularity organization (Ravasz *et al.* 2002) Ravasz *et al*. proposed a hierarchical modularity model for metabolic networks. According to this model, metabolic networks of organisms are organized as many small, but highly connected modules that combine in a hierarchical manner to larger, less cohesive units. Several recent studies using concepts

such as the reaction betweenness centrality distribution and the dependency of metabolites have further verified that metabolic networks are organized in a hierarchical way (Holme *et al.* 2003; Gagneur *et al.* 2003). These results indicate that hierarchical modularity is also an important feature of metabolic networks. Modularity has been shown to be common in the organization of robust and sustainable complex systems (Hartwell *et al.* 1999). Therefore, identifying the modular organization of metabolic network by certain network decomposition methods can help us in better understanding the organization principle of complex systems.

A possible large-scale design principle is that one part (module) of the network constitutes a densely connected core that also is central in terms of network distance, and the rest of the network forms a periphery (Holme 2005). In, for example, a network of airline connections one would most certainly pass such a core-airport on any many-flight itinerary. In metabolic networks, the core part is the central metabolism where many metabolites will be converted to supply the necessary metabolites to other, less central, pathways (modules in the periphery part).

1.2.2 Networks used in this work

For this work, the metabolic networks of *Aeropyrum pernix*, *Bacillus subtilis*, *Escherichia coli*, *Saccharomyces cerevisiae* and *Homo sapiens* were used. These networks originated from the work of reconstruction of metabolic networks from genome data from Ma and Zeng (Ma & Zeng 2003).

Random networks were generated for comparison purpose when a network not presenting a core-periphery structure was required (see Chapter 5). For the computer-generated networks, two models of random networks were used. A Barabási-Albert (Barabasi & Albert 1999; Jeong *et al.* 2000) network and a Newman-Watts-Strogatz network (Newman *et al.* 2002). Both networks were generated with 500 nodes (the average of the real-data networks for better comparison). The average number of neighbors of each node (average degree – see Section 1.3.2) in each network is similar to the value in the metabolic networks studied, which is between 2 and 3, to make the comparison of the random networks and the metabolic networks more meaningful.

1.3 Graph Theory

In this work terms from graph theory like network, nodes and edges are used to describe the algorithms involved in the network analysis. Here a short introduction to graph theory is given to explain these terms.

The term network is an informal description for a set of elements with connections or interactions between them (Junker & Schreiber 2006). A typical example from biology is a metabolic network. It consists of a set of metabolites (elements) and a set of interactions between them (biochemical or chemical reactions, also called connections).

To deal with networks in a formal way they are modeled as graphs. A graph is a mathematical object consisting of vertices and edges representing elements and connections, respectively. This usage of the term "graph" should not be confused with another meaning often used in biology: the graphical representation of a function in the form of a curve or surface.

1.3.1 Basic notation

A graph $G = (V, E)$ consists of a set of vertices (nodes, points) V and a set of edges (arcs, lines) E, where each edge is assigned to two vertices. An edge e connecting the vertices u, v is denoted by $\{u, v\}$ and we say u and v are *incident* with e and *adjacent* or *neighbors* to each other. The vertices incident to an edge are called its *end-vertices*. Figure 1.2 shows a sample graph.

1.3.2 Degree

The degree of a vertex v is the number of edges that have v as end-vertex. An edge where the two end-vertices are the same vertex is called a *loop*. In the graph from Figure 1.2, node **1** has degree equals to 2 and node **6** has degree equals to 3.

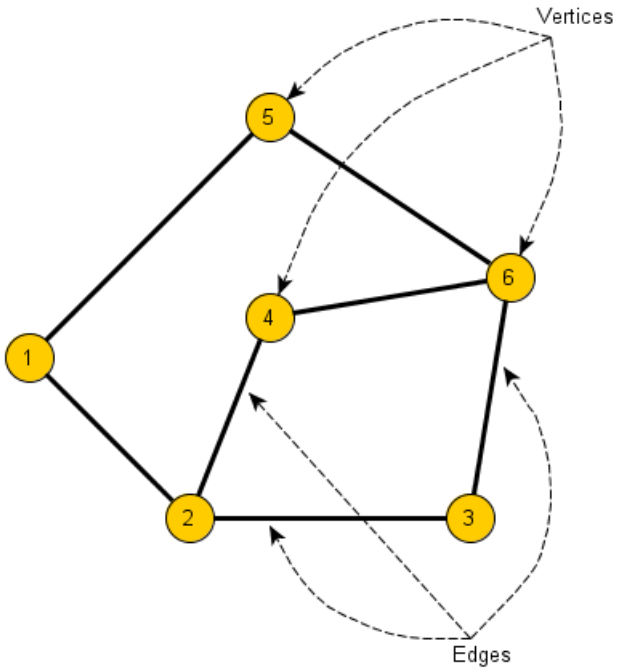


Figure 1.2 – Sample graph

1.3.3 Path

A path in a graph is a sequence of vertices such that from each of its vertices there is an edge to the successor vertex. The first vertex is called the start vertex and the last vertex is called the end vertex. Both of them are called end or terminal vertices of the path. The other vertices in the path are internal vertices. In the graph from Figure 1.2 from node 1 to node 6 there are 3 possible paths: 1→5→6; 1→2→3→6 and 1→2→4→6.

1.3.4 Shortest path

The shortest path is a path between two vertices such that the sum of the weights of its constituent edges is minimized. In an unweighted graph, where all the weights are equal to 1, the shortest path between two vertices is simply the path with the smallest number of edges to connect the two vertices. In the graph from Figure 1.2 the shortest path from node 1 to node 6 is 1→5→6.

1.3.5 Distance

The distance between two vertices in a graph is the number of edges in a *shortest path* connecting them. This is also known as the *geodesic distance*. In the graph from Figure 1.2 the distance between nodes 1 and 6 is equal to 2.

1.3.6 Eccentricity

The eccentricity ε of a vertex *v* is the *greatest distance* between *v* and any other vertex. In the graph from Figure 1.2 all the nodes have eccentricity equals to 2.

1.3.7 Radius

The radius of a graph is the *minimum* eccentricity of any vertex. The radius of the graph from Figure 1.2 is 2 because all the nodes have eccentricity equals to 2.

1.3.8 Diameter

The diameter of a graph is the *maximum* eccentricity of any vertex in the graph. That means, it is the greatest distance between any two vertices. This is equivalent to say that the diameter of a network is defined as the longest of all the shortest pathways in the network. In the case of the graph from Figure 1.2 the diameter is equals to the radius because all nodes have the same eccentricity.

1.3.9 Centrality

A method to understand networks and their participants is to evaluate the location of the nodes in the network. Measuring the node location is finding the *centrality* of this node. This measure helps determine the importance, or prominence, of a node in the network.

Figure 1.3 shows a social network called *Kite network* (developed by David Krackhardt) (Krebs 2006) where different types of centralities can be seen.

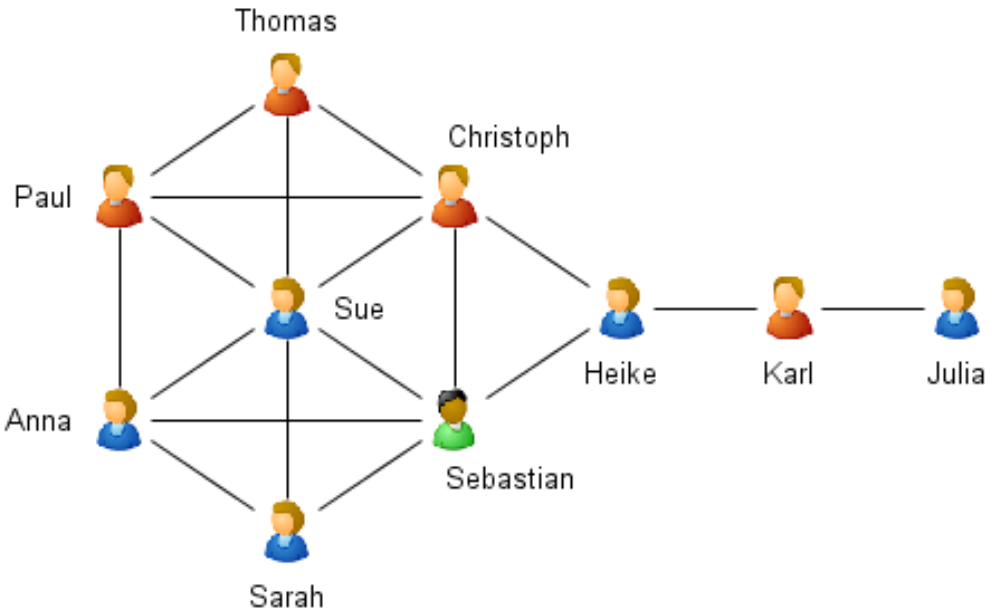


Figure 1.3 – *Kite* network

This network is a social network that shows relations between 10 people. In this network, two people are connected if they regularly talk to each other, or interact in some way. For instance, in the network above, Paul regularly interacts with Thomas, but not with Karl. Therefore Paul and Thomas are connected, but there is no link drawn between Paul and Karl. This network clearly shows the distinction between the three most popular individual network measures: *degree centrality*, *betweenness centrality*, and *closeness centrality* which are further explained below. Table 1.1 shows the values of the different types of centrality for the nodes in the *Kite* network.

1.3.9.1 Degree centrality

Social network researchers measure network activity for a node by using the concept of degrees — the number of direct connections a node has. In the *Kite network* mentioned above, Sue has the most direct connections in the network, making hers the most active node in the network. She is a “connector” or “hub” in this network. Common wisdom in personal networks is that “the more connections, the better.” This is not always so. What really matters is where those connections lead to — and how they connect the otherwise

unconnected. Here Sue has connections only to others in her immediate cluster. She connects only those who are already connected to each other.

Table 1.1 – Values of centrality for the *Kite* network

a)	betweeness centrality
Heike	0.388889
Christoph	0.231481
Sebastian	0.231481
Karl	0.222222
Sue	0.101852
Paul	0.023148
Anna	0.023148
Sarah	0.000000
Thomas	0.000000
Julia	0.000000

b)	closeness centrality
Christoph	0.642857
Sebastian	0.642857
Sue	0.600000
Heike	0.600000
Paul	0.529412
Anna	0.529412
Sarah	0.500000
Thomas	0.500000
Karl	0.428571
Julia	0.310345

c)	degree centrality
Sue	0.666667
Christoph	0.555556
Sebastian	0.555556
Paul	0.444444
Anna	0.444444
Sarah	0.333333
Thomas	0.333333
Heike	0.333333
Karl	0.222222
Julia	0.111111

In other words, degree centrality measures the connectivity of the nodes based on the number of direct connections a node has.

The degree centrality of a node is calculated as the fraction of nodes that are connected to each node. So:

$$C_D(n) = \frac{d_n}{N-1} \qquad (1.1)$$

where d_n is the number of links (degree) of node n and N is the number of nodes in the network.

1.3.9.2 Betweeness centrality

While Sue has many direct ties, Heike has few direct connections — less than the average in the network. Yet, in many ways, she has one of the best locations in the network — she is between two important constituencies. She plays a “broker” role in the network. She plays a powerful role in the network but she is also a single point of failure. Without her, Karl and Julia would be cut off from information and knowledge in Sue's cluster. A node with high betweenness has a great influence over what flows in the network.

Betweenness centrality (Brandes 2000) measures how many shortest pathways in the network pass trough a node. A bigger number of paths through the node means a higher importance of the node.

The betweenness centrality is calculated as the fraction of number of shortest paths that go through each node:

$$C_B(n) = \frac{\sigma_n}{\sigma} \tag{1.2}$$

where σ_n is the number of shortest paths that go through node *n* and σ is the number of (shortest) paths of the network.

1.3.9.3 Closeness centrality

Christoph and Sebastian have fewer connections than Sue, yet the pattern of their direct and indirect ties allow them to access all the nodes in the network more quickly than anyone else. They have the shortest paths to all others — they are close to everyone else. They are in an excellent position to monitor the information flow in the network — they have the best visibility into what is happening in the network.

With closeness centrality, the "position" of a node in the network is measured. The most central node is the one with the shortest path to other nodes in the network. Nodes at the periphery have a long path to nodes in the opposite side of the network.

The closeness centrality of a node is measured as the reciprocal of the average distance from this node to all other nodes in the network (Brandes 2000):

$$C_C(n) = \frac{1}{\overline{d_n}} \tag{1.3}$$

where $\overline{d_n}$ is the average distance of node *n* to other nodes in the network.

1.3.9.4 Peripheral Players

Most people would view the nodes on the periphery of a network as *not* being very important. In fact, Karl and Julia receive very low centrality scores for *this* network. Yet, peripheral nodes are often connected to networks that are not currently mapped. Karl and Julia may be contractors or vendors that have their own network outside of the company — making them very important resources for fresh information not available inside this same company.

In a biological network, these nodes can connect sub-networks. In the case of an integrated network, these nodes may connect different networks that are integrated like metabolic networks, regulatory networks, protein-protein interaction networks, etc.

1.3.9.5 Network Centralization

Individual network centralities provide insight into the individual's location in the network. The relationship between the centralities of all nodes can reveal much about the overall network structure.

A very centralized network is dominated by one or a few very central nodes. If these nodes are removed or damaged, the network quickly fragments into unconnected sub-networks. A highly central node can become a single point of failure. A network centralized around a well connected hub (node with high degree and betweenness centrality) can fail abruptly if that hub is disabled or removed.

A less centralized network has no single points of failure. It is more resistant in the face of many intentional attacks or random failures – many nodes or links can be removed while allowing the remaining nodes to still reach each other over other network paths. Networks of low centralization fail gracefully.

1.3.9.6 Centrality in metabolic networks

Centrality measurements can help to identify important nodes in the network. Degree centrality can show the nodes with highest number of connections in the network. This

helps to identify the “hubs” in the network. In a metabolic network, this means the nodes that can be converted into more metabolites.

Betweenness centrality may also help to find important nodes in the network. The node with the highest betweenness centrality is the one with the highest number of shortest pathways going through it. For a metabolic network, this may means nodes that participate in many metabolites conversions. This is however not always true as it is shown in Section 5.4.1.2.

With closeness centrality, one can identify nodes that are more central in the network and nodes in the periphery part. Nodes in the central part of the network have a shorter distance to other nodes in the network (they are closer to other nodes, therefore the name “closeness”). In a metabolic network, this means metabolites that can be converted into other metabolites in the shortest number of steps.

Tables 1.2 to 1.4 show the top 10 nodes by different centrality measures for the metabolic network from *B. subtilis*.

Table 1.2 – 10 metabolites with the highest betweenness centrality for *B. subtilis* network.

pos	betwenness centrality	KEGG	Name
1	0.432132	C00022	Pyruvate
2	0.310035	C00117	D-Ribose 5-phosphate
3	0.297295	C00119	5-Phospho-alpha-D-ribose 1-diphosphate
4	0.282609	C00118	D-Glyceraldehyde 3-phosphate
5	0.278441	C03090	5-Phosphoribosylamine
6	0.275247	C03838	5'-Phosphoribosylglycinamide
7	0.272031	C04376	5'-Phosphoribosyl-N-formylglycinamide
8	0.268792	C04640	2-(Formamido)-N1-(5'-phosphoribosyl)acetamidine
9	0.265530	C03373	Aminoimidazole ribotide
10	0.262246	C04751	1-(5-Phospho-D-ribosyl)-5-amino-4-imidazolecarboxylate

Table 1.3 – 10 metabolites with the highest closeness centrality for *B. subtilis* network.

pos	closeness centrality	KEGG	Name
1	0.124260	C00022	Pyruvate
2	0.122129	C04442	2-Dehydro-3-deoxy-6-phospho-D-gluconate
3	0.122058	C11437	1-Deoxy-D-xylulose 5-phosphate
4	0.122022	C00074	PEP
5	0.121247	C00118	D-Glyceraldehyde 3-phosphate
6	0.120309	C04691	DAHP
7	0.119590	C00036	OAA
8	0.118243	C00279	E4P
9	0.115543	C00111	Glycerone phosphate
10	0.115195	C00085	D-Fructose 6-phosphate

Table 1.4 – 10 metabolites with the highest degree centrality for *B. subtilis* network.

pos	degree centrality	KEGG	Name
1	0.038095	C00022	Pyruvate
2	0.026190	C00024	Acetyl-CoA
3	0.023810	C00025	Glutamate
4	0.021429	C00092	D-Glucose 6-phosphate
5	0.019048	C00085	D-Fructose 6-phosphate
5	0.019048	C00124	D-Galactose
5	0.019048	C00111	Glycerone phosphate
5	0.019048	C00118	D-Glyceraldehyde 3-phosphate
9	0.016667	C00031	D-Glucose
9	0.016667	C00248	Lipoamide
9	0.016667	C00049	Aspartate

1.3.10 Capacity

Capacity (Ma, unpublished work) is a property of a network to measure its robustness. It is defined as:

$$C = \sum_{i=1}^{n} \frac{1}{PL_i} \tag{1.4}$$

where n is the total number of connected pairs in the network and PL_i is each one of the shortest path lengths.

By measuring the capacity of a network before and after the removal of some nodes one can measure the robustness of this network. This can be done by a simply comparison of the capacity with and without that nodes. A network with more paths will have a higher capacity. In the same way, if the paths in the network are shorter the capacity will be higher.

1.3.11 Fragility

Fragility of a node (Ma, unpublished work) measures how much the capacity of the network will be affected by the removal of that node and it is defined as:

$$f(n) = 1 - \frac{C_n}{C_N} \tag{1.5}$$

where C_N is the capacity of the whole network and C_n is the capacity of the network without node n. Fragility ranges from 0 to 1.

1.3.12 Modularity coefficient

When working with network decomposition, one needs to find ways to measure the quality of the decomposition. Newman (Newman & Girvan 2004) proposed a measure called *modularity* for this.

A good partition of a network into modules must comprise many within-module links and as few as possible between-module links. However, if we just try to minimize the number of between-module links (or, equivalently, maximize the number of within-module links) the optimal partition consists of a single module and no between-module links.

Modularity is defined as (Newman & Girvan 2004; Guimera & Nunes Amaral 2005):

$$M = \sum_{s=1}^{N_M} \left[\frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right] \tag{1.6}$$

where N_M is the number of modules, L is the number of links in the network, l_s is the number of links between nodes in module s and d_s is the sum of the degrees of the nodes in module s.

If the number of within-communities is no better then random, we will get $M = 0$. Values approaching $M = 1$, which is the maximum, indicate strong community structure. Values for social networks studied by Newman fall in the range from about 0.3 to 0.7 if there is a community structure present (Newman 2004). From our experience this value for metabolic networks is around 0.8 or higher (see tables 4.2 and 4.3), which shows the strong community structure in this kind of networks.

1.3.13 Special Graphs

There are many different biological networks with different properties. Often the graph model has to be tailored to the specific network under consideration.

1.3.13.1 Undirected, directed and mixed graphs

Graphs can be undirected, directed or mixed. In an undirected graph an edge between the vertices u and v is represented by the unordered vertex pair $\{u, v\}$. A typical example from biology is the protein-protein interaction network. Figure 1.2 shows an example of an undirected graph.

In a directed graph an edge between the vertices u and v is represented by the ordered vertex pair (u, v). In visualization of graphs the direction of an edge is usually represented by an arrowhead. The two edges (u, v) and (v, u) between the vertices u, v can be either represented by two lines or by one line with arrowheads at both ends. Typical examples of biological networks modeled by directed graphs are metabolic network and gene regulatory network. Figure 1.4(a) shows an example of directed graph.

(a) (b)

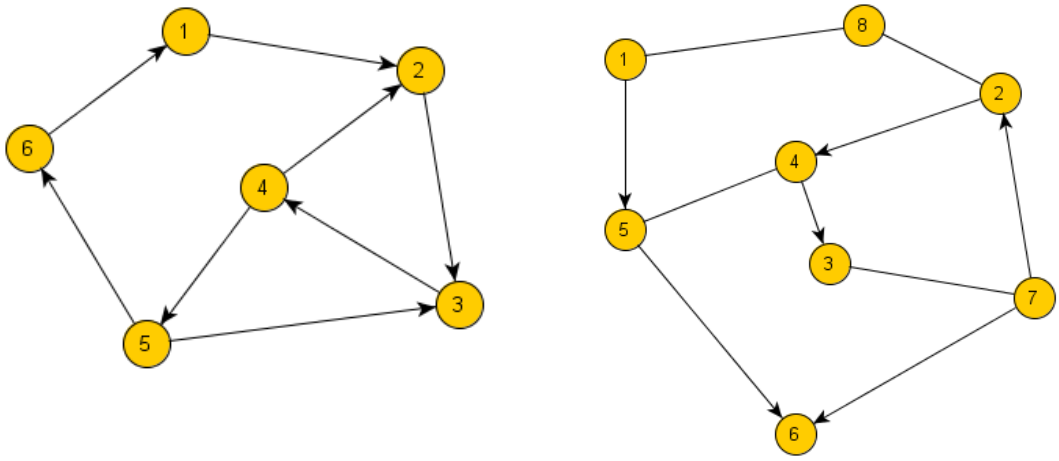


Figure 1.4 – (a) Directed graph; (b) mixed graph.

In a mixed graph both undirected and directed edges occur. This type of graph is also relevant to biology. An example for this is special protein-protein interaction networks in which some of the interactions are undirected (e. g. obtained by two-hybrid experiments) and others are directed representing activation, phosphorylation and other directed interactions. Also in integrated cellular networks mixed graph can be used. Figure 1.4(b) shows an example of mixed graph.

An undirected edge has end-vertices; a directed edge (u,v) has a source vertex u (origin, head) and a target vertex v (destination, tail). In a directed graph a vertex has an out-degree which is defined as the number of edges going out of it and an in-degree defined as the number of edges coming into it. The degree of the vertex is the sum of its in- and out-degrees.

Whether the direction in the network is considered or not will affect the calculation of the shortest path because for the calculation of the shortest path in a directed network, the edges should be followed according to their direction. For example, in Figure 1.4(a), to reach vertex 3 from vertex 1, the shortest path considering the direction of the edges is 1→2→3. From vertex 3 to vertex 1, the shortest path is 3→4→5→6→1. It is longer because the paths should respect the direction of the edges.

1.3.13.2 Hyper-graphs and bipartite graphs

There are biological networks where more than two elements are connected by an interaction. For instance, in metabolic networks many biochemical reactions have more than one substrate and more then one product. To model such networks, hyper-graphs are used. In hyper-graphs, the edges are called hyper-edges because they can interconnect more than 2 vertices at the same time (see Figure 1.5(a)). Similar to graphs, hyper-graphs can also be defined as directed, undirected or mixed graphs.

Hyper-graphs are not commonly used in graph theory and many algorithms developed for graphs cannot be directly applied to hyper-graphs. Therefore, hyper-graphs are seldom used to model biological networks. Instead these networks are modeled by *bipartite graphs*, a structure generally used to represent hyper-graphs. In a bipartite graph for each hyper-edge there is an intermediate vertex that represents this hyper-edge. This vertex connects all the vertices connected by the hyper edge. In case of metabolic networks this intermediate vertex represents the reaction (or the enzyme) as shown in Figure 1.5(b). Figure 1.5 shows an example of representing metabolic pathways with a hyper-graph and with a bipartite graph.

The networks studied in this work use an even simpler representation method as described by Ma and Zeng (Ma & Zeng 2003). All the networks used here are originated from the work of Ma and Zeng, because it was not the purpose of this work to do the reconstruction of the networks. It was also desired to compare the results with other works (Guimera & Nunes Amaral 2005; Ma *et al.* 2004) that have used the same networks. Using this simpler representation, the pathway from Figure 1.5 is represented as shown in Figure 1.6 if the currency metabolites are considered. If the currency metabolites (**NAD**, **Pi** and **NADH**) are removed, this pathway is simplified to a single link between **Glyceraldehyde 3-P** and **1,3-P Glycerate**.

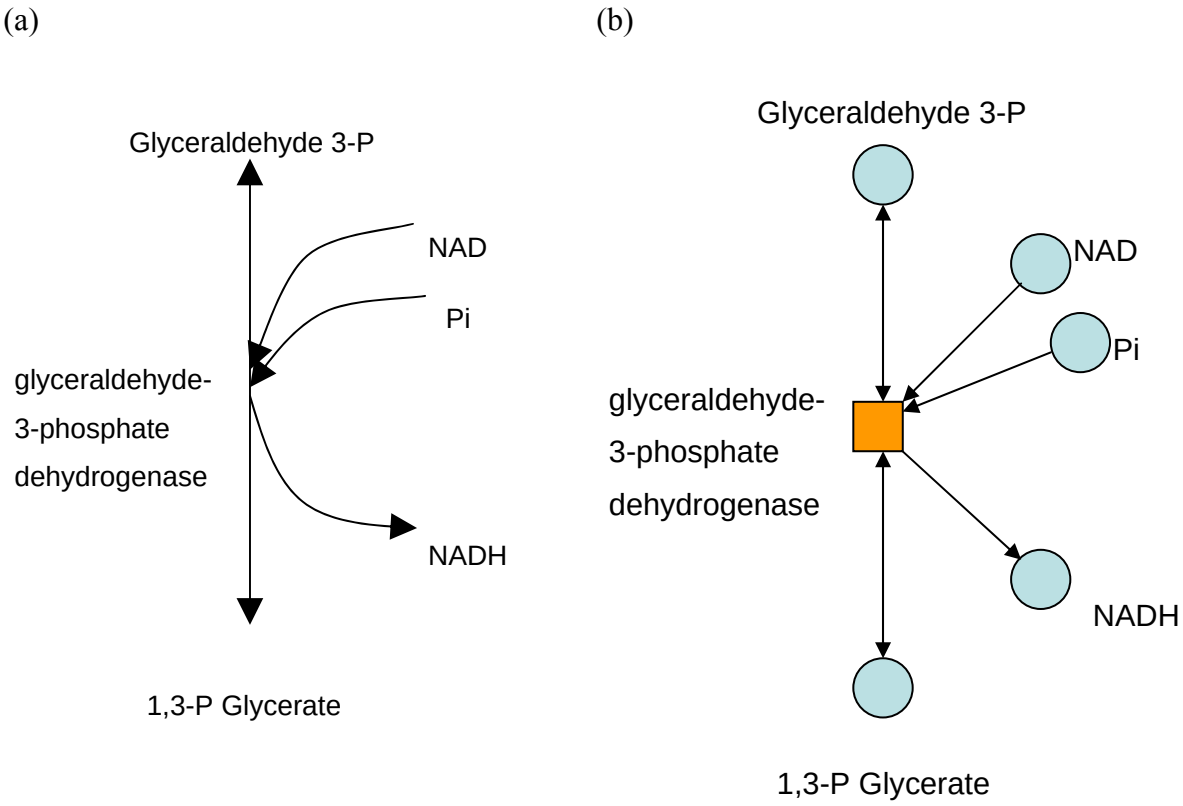


Figure 1.5 – A metabolic pathway and its modeling as a bipartite graph.
(a) The pathway modeled with a hyper-graph. (b) The same pathway modeled as a bipartite graph.

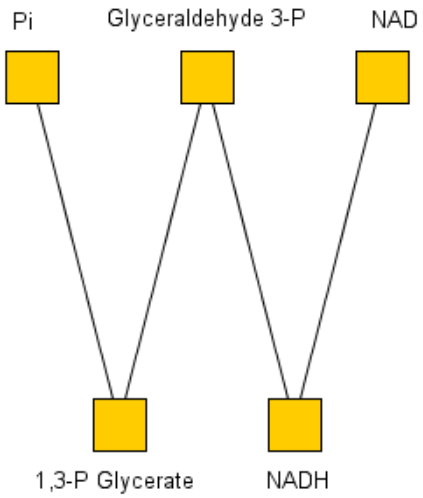


Figure 1.6 – Representation as a normal graph.

Chapter 2
Tools for network analysis and visualization

Understanding and analyzing complex networks can be very difficult, if not impossible, without the help of good tools to analyze and visualize this data. There are lots of such tools available. Some of them are free and others commercial. During this work, we tried to make use of the available tools. However, we often face to the situation that no available tools could help us to solve our specific questions. It is urgent to develop new tools (or extend the current ones) to help biologist to handle the rapidly emerging genome-wide large-scale complex networks.

In this chapter, tools for visualization and analysis of networks developed in this work are described. Some of the tools were developed from scratch to meet the special needs of the research while others are extensions to expand existing tools like *Cytoscape*.

Table 2.1 lists the tools and languages used to develop them. It also shows the size of the tools developed (total number of lines not including external libraries) which could be rough measure of the complexity of the tools. Because of the big amount of code, the listings of the programs are not included in the written work (but are available on request or in the website of the tools).

Table 2.1 – Languages used to develop the tools.

Application	Language	lines
NetConv	Tcl/Tk	356
Cytoscape plugins	Java	2813
MOS	Python	1012
pyNetConv	Python	1571
cluster tool (v1)	Python	5213
libclust	Python	7308
Total		18273

2.1 Introduction

First a brief introduction of the languages and software used to develop the software tools described later in this chapter. A description of *Cytoscape* is given as parts of the software developed in this work are *Cytoscape* plugins.

2.1.1 Cytoscape

Cytoscape (http://cytoscape.org) (Shannon *et al.* 2003) is a bioinformatics software platform for visualizing molecular interaction networks and integrating these interactions with gene expression profiles and other biological data.

Cytoscape supports the following features:

Input

- Input and construct molecular interaction networks from raw interaction files (SIF format) containing lists of protein-protein and/or protein-DNA interaction pairs. For yeast and other model organisms, large sources of pairwise interactions are

available through the BIND and TRANSFAC databases. User-defined interaction types are also supported.

- Load and save previously-constructed interaction networks in GML format (Graph Markup Language).
- Input mRNA expression profiles from tab- or space-delimited text files.
- Load and save arbitrary attributes on nodes and edges. For example, input a set of custom annotation terms for proteins; create a set of confidence values for protein-protein interactions.
- Import gene functional annotations from the Gene Ontology (GO) and KEGG databases.

Visualization

- Customize network data display using powerful visual styles.
- View a superposition of gene expression ratios and *p-values* on the network. Expression data can be mapped to node color, label, border color, etc. according to user-configurable colors and visualization schemes.
- Layout networks in two dimensions. A variety of layout algorithms are available, including cyclic and spring-embedded layouts.
- Zoom in/out and pan for browsing the network.
- Use the network manager to easily organize multiple networks.

Analysis

Plugins are available for network and molecular profile analysis. For example:

- Filter the network to select subsets of nodes and/or interactions based on the current data. For instance, users may select nodes involved in a threshold number

of interactions, nodes that share a particular GO annotation, or nodes whose gene expression levels change significantly in one or more conditions according to *p-values* loaded with the gene expression data.

- Find active subnetworks / pathway modules. The network is screened against gene expression data to identify connected sets of interactions, i.e. *interaction subnetworks*, whose genes show particularly high levels of differential expression. The interactions contained in each subnetwork provide hypotheses for the regulatory and signaling interactions in control of the observed expression changes.

- Find clusters (highly interconnected regions) in any network loaded into Cytoscape. Depending on the type of network, clusters may mean different things. For instance, clusters in a protein-protein interaction network have been shown to be protein complexes and parts of pathways. Clusters in a protein similarity network represent protein families.

Cytoscape was initially made public in July, 2002 (v0.8); the second release (v0.9) was in November, 2002 and v1.0 was released in March 2002. Version 1.1.1 is the last stable release for the 1.0 series. Version 1.1.1 has some of the features listed above and some additional features, such as the ability to add and remove nodes and undo actions.

Figure 2.1 shows a screenshot of *Cytoscape* running with *yeast* data loaded.

Cytoscape is open-source software. It is possible to extend *Cytoscape* functionality by developing plugins. *Cytoscape* plugins are special software developed separately that can be load into *Cytoscape* to add new features. With plugins it is possible to use all the functions available in *Cytoscape* without the need to write a full application from scratch. Only the code for the new functionalities needs to be written.

To see some example of Cytoscape plugins, look at http://cytoscape.org/plugins2.php .

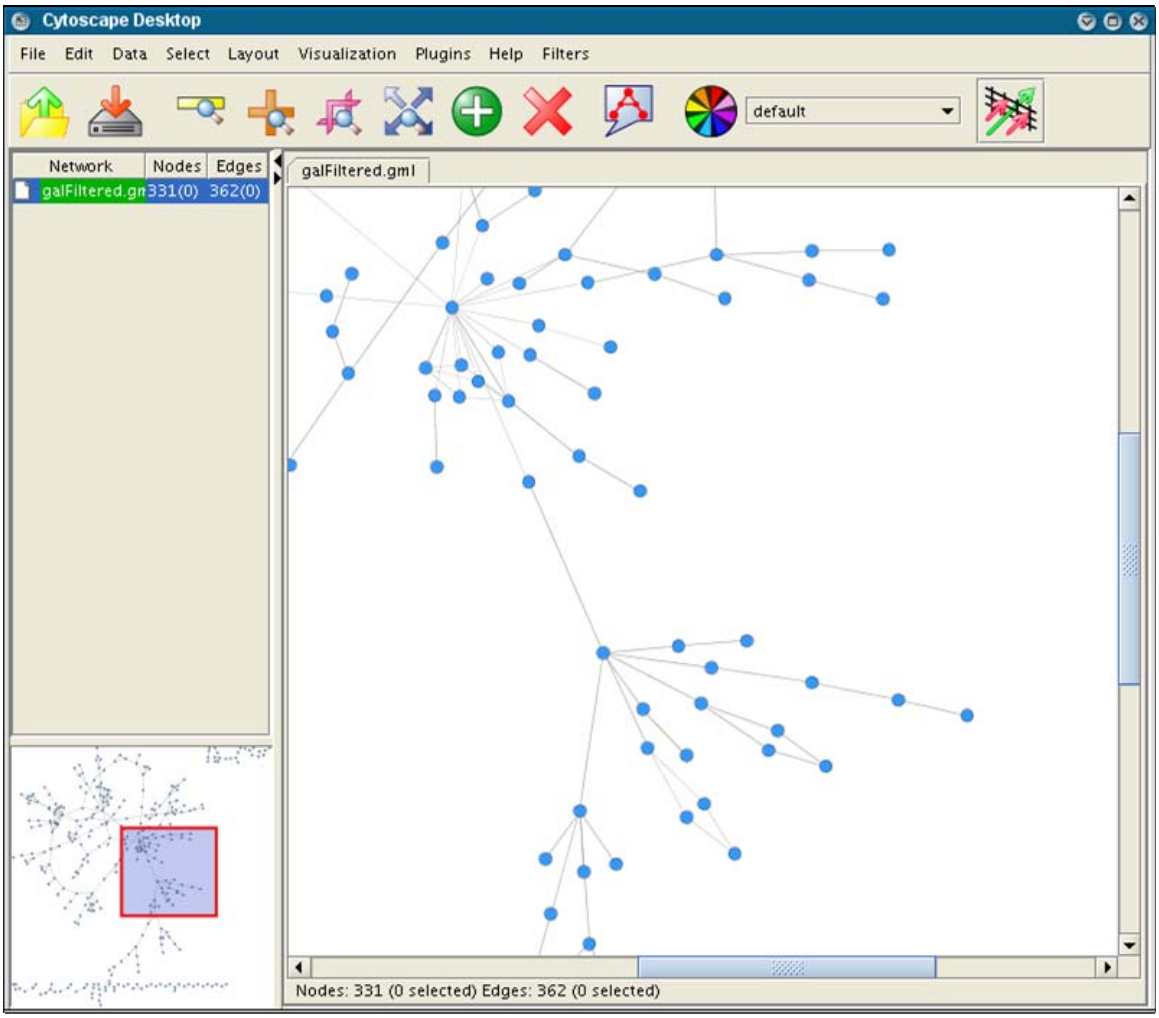


Figure 2.1 – Cytoscape main window

2.1.1.1 Cytoscape file formats

Here a brief description of some of the Cytoscape files formats used in this work. For more information on this formats and another types of files supported by Cytoscape, see the manual available online at http://cytoscape.org .

The formats for network supported by Cytoscape are the SIF format and GML. GML is a more complete format, supporting not only network structure but also other attributes like node position. The problem in generating GML manually is that the file is much more complicated then files in SIF format. The SIF format is very simple and easy to create.

The drawback is that it can only represent the network interactions without any other information like node position. So, after loading a file in SIF format, the user needs to apply one of the layouts available in Cytoscape to have a better view of the network, since the default for SIF format is to show the nodes in a rectangular matrix.

SIF file format is a text file. Its format is simply a list of interactions and, for isolated nodes, a list of nodes. In the SIF file, each line represents one interaction or one node. For isolated nodes, only the node should appear on the line; for interactions, there are 3 columns to be entered per line separated by space or tab: source node, interaction type and target node. The interaction type can be any text, and is only used to differentiate the types of reactions existent in the network. Some conventions are used but not obligated. Table 2.2 shows the convention for interaction types. Listing 2.1 shows an example of a network in SIF format.

Table 2.2 – Convention used for SIF file interaction type

Symbol	Interaction type
pp	protein – protein interaction
pd	protein -> DNA (e.g. transcription factor binding upstream of a regulating gene.)
pr	protein -> reaction
rc	reaction -> compound
cr	compound -> reaction
gl	genetic lethal relationship
pm	protein-metabolite interaction
mp	metabolite-protein interaction

Listing 2.1 – Example of SIF file format

```
A pp B
B pp C
A pp C
D
```

In the example of Listing 2.1, a network with 5 nodes (A, B, C, D and E) is defined and interactions between nodes A, B and C exist.

If the user has a file in other format, *pyNetConv* (see Section 2.3.1.2) may be used to convert it to Cytoscape format.

Another type of Cytoscape file used in this work is the node attributes file. Like the SIF file, this one is very simple to create. The first line is the attribute name and each of the other lines has the following format:

```
node = attribute
```

where node is the node name as used in the SIF file and the attribute is the desired attribute to add to the node. Attributes can be numeric or text. Listing 2.2 shows an example of node attributes file where the attribute shows module each node belongs to.

Listing 2.2 – Example of Cytoscape node attributes file

```
Modules
A = module1
B = module1
C = module2
D = module3
E = module2
```

There is also a file for entering expression values like the ones generated by microarray experiments. This file has the following format:

```
GeneName [CommonName] ratio1 ratio2 ... ratioN [pval1 pval2 ... pvalN]
```

The first two fields are the gene name and an optional common name. Expression ratios are provided for each experiment, optionally followed by a p-value per experiment or other measure of the significance of each ratio, i.e. whether the ratio represents a true change in expression (according to some statistical model.)

2.1.2 Python

Python (http://python.org) is an *interpreted, interactive, object-oriented* programming language.

Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC, wxWidgets). New built-in modules are easily written in C or C++. *Python* is also usable as an extension language for applications that need a programmable interface.

The Python implementation is portable: it runs on many brands of UNIX, on Windows, OS/2, Mac, Amiga, and many other platforms. For platforms were Python was not ported yet, there is a version of the language written in Java called Jython (http://www.jython.org) which will run in any platform supported by the Java language. There is also a recent port of Python called IronPython (http://www.ironpython.com) that supports the new Microsoft .NET platform (http://www.microsoft.com/net). That means, software written in Python can run in different platforms. This was the case of the tools developed in this work were the development platform was Linux but the software is used by many Windows users.

Python was chose for the development of some of the tools in this work because it is very good for creating prototypes of programs (thanks to its dynamic nature and the big number of libraries available). After the first prototype is developed and the algorithm is stable, it should not be too hard to port the software to another language. Porting may not be necessary as many of the Python libraries are developed in C. A hybrid solution is also possible: after identifying the functions of the software developed in Python that need to be speed up, it is possible to recode only this part in another language (like C) and keep the rest of the software untouched. This hybrid approach combines the better of 2 worlds: fast development (Python) and fast program execution speed (C).

2.1.3 Tcl/Tk

Tcl/Tk (http://tcl.tk) is a script language developed by Prof. John Ousterhout in the University of California at Berkeley.

Like Python, Tcl/Tk is a cross-platform language and it is very easy to learn. It is very well suited for development of graphical interfaces to other software as it can be easily embedded or extended to fit the requirements of the application. Its graphic toolkit (the Tk part in Tcl/Tk) was ported to many other languages (including Python) because of its easy of use and portability.

While it is very easy and fast to create applications in Tcl/Tk, the language itself is not very fast and, because of this, not too much used in applications that require much processing. Similar to Python it is possible to extend the language in C/C++ to increase speed.

2.1.4 Java

Java is an object-oriented programming language developed by James Gosling and colleagues at Sun Microsystems in the early 1990s. The language, which was designed to be platform independent, is a derivative of *C++* with a simpler syntax, a more robust runtime environment and simplified memory management.

Java is not related to *JavaScript*, though they have similar names and share a *C*-like syntax.

2.2 Cytoscape plugins

During this work, several plugins for *Cytoscape* were developed. These plugins are available on Internet at Source Forge (http://www.sourceforge.net). Source Forge is a project that hosts open source projects for free, offering many resources to the developers. The home of the plugins described in this work at Source Forge is at http://csresources.sourceforge.net .

2.2.1 PAEContext plugin

Cytoscape is developed by a group working mainly with *yeast*. Because of this, most of the built in support in *Cytoscape* is for *yeast*.

While working with *P. aeruginosa*, the need to visualize this data aroused. As we use Cytoscape for visualization, this plugin was created to add functionality to Cytoscape to work with *P. aeruginosa*.

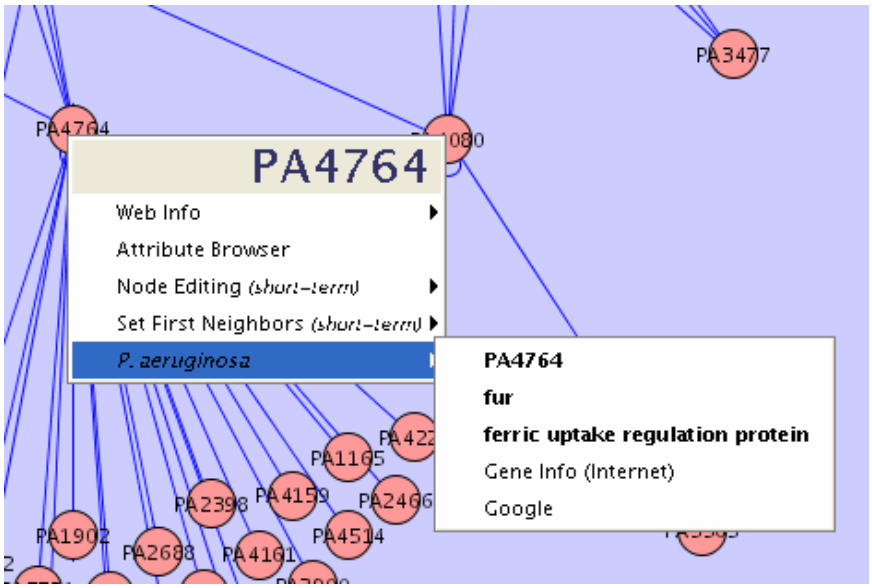


Figure 2.2 – PAEContext plugin

This plugin adds to *Cytoscape* a menu with some basic information about the selected gene like the gene name and its product as shown in Figure 2.2. The plugin has also options to search more information about the gene on Internet, either at http://www.pseudomonas.com (option **`Gene Info (Internet)`** of the menu) or using the Google search engine.

2.2.2 ShortestPath plugin

Sometimes, while analyzing a network, it is interesting to know the shortest path between two nodes. In a metabolic network this means the shortest number of reactions required to reach a metabolite from another one.

With this plugin it is possible to visualize in *Cytoscape* the shortest path between 2 nodes. The Dijkstra algorithm (Jungnickel 2002; Ikeda 2000) was used in this plugin.

To find the shortest path between two nodes, the user needs to select two nodes in the network and call the plugin by selecting it from the menu. There are two options in the menu for the shortest path plugin as shown in Figure 2.3.

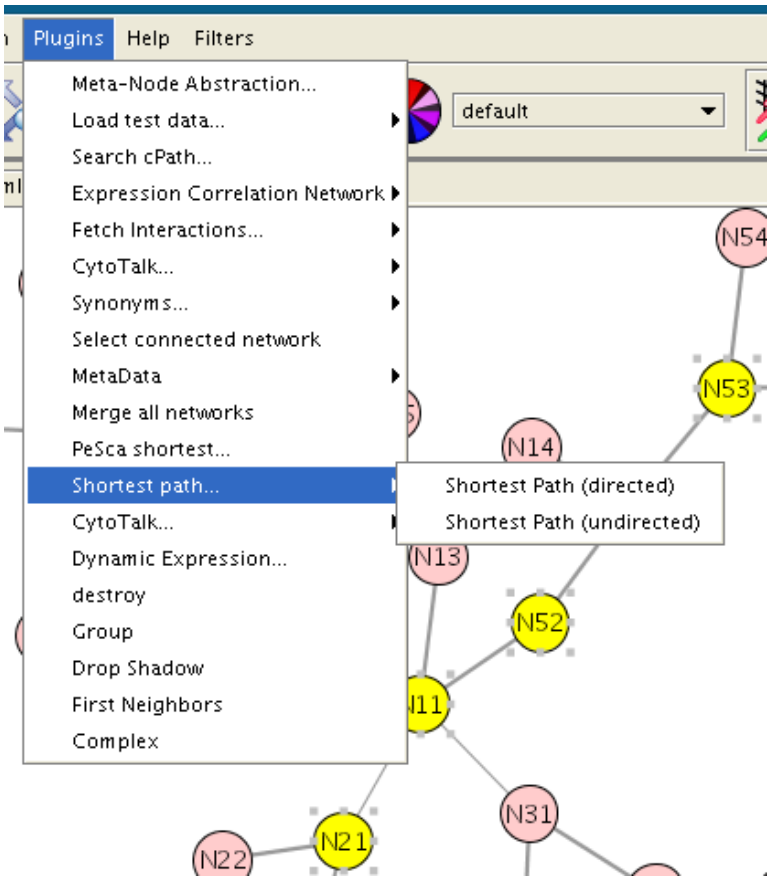


Figure 2.3 – ShortestPath plugin menu

If the user chooses the *undirected* version, the plugin will find the shortest path between the two nodes without taking into account the direction of the edges (for irreversible reactions, for example). If *directed* is chosen, then the plugin will first confirm the order of the two nodes should to be considered (as source and target) and then it will shows the shortest path between the two nodes by considering the direction of the edges. As an

example, Figure 2.4 shows the effects of calling the plugin, after selecting nodes N51 (as the source node) and N24 (as the target node), using directed and undirected options.

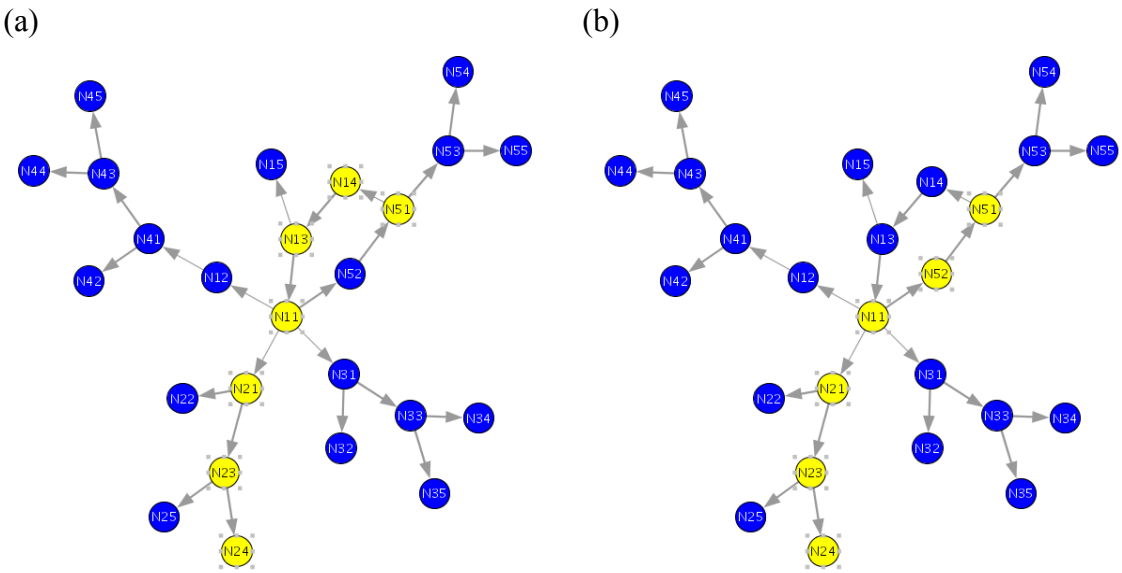


Figure 2.4 – Use of ShortestPath plugin
(a) Shortest path considering direction; (b) shortest path ignoring direction

If, in the example from Figure 2.4, we had selected directed shortest path but used N24 as the source node and N51 as the target node, the plugin would inform that there is no path between both nodes. In a metabolic network, this would show that there is no way to convert from metabolite N24 to metabolite N51.

2.2.3 SelConNet plugin

While working with a network it may be desired to have only the connected part of the network for further analysis. This plugin was created to help this task.

This plugin will allow one to extract a connected part of a network directly in a *Cytoscape* session. The user needs to select one or more nodes from the desired connected part to be extracted and call the plugin from the menu.

After the plugin execution, the desired connected part will be selected and the user has the option to extract it and save it separately. Figure 2.5 shows an example for metabolic data from *E.coli*.

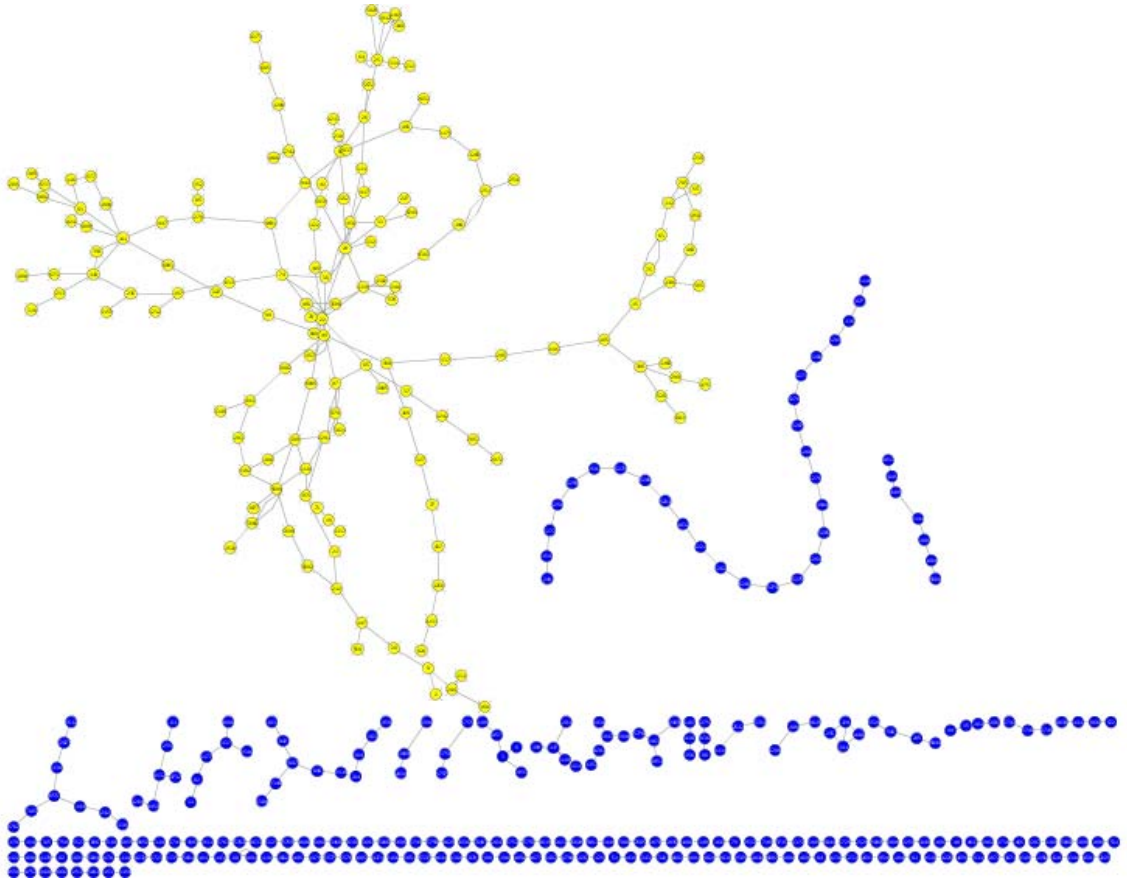


Figure 2.5 – SelConNet plugin in action

In the example of Figure 2.5, some nodes of the biggest connected part of the metabolic network from *A. pernix* were selected, and after the plugin was called, all the nodes belonging to this connected part of the network were selected (shown in yellow in the figure).

After this step, to extract the selected nodes to a new network one just need to use the built-in Cytoscape function for this task.

2.2.4 MetaData plugin

This plugin works together with the cluster tool (see Section 2.3.2) and can be used for visualization of the generated modules.

The main objective of this plugin is to show in Cytoscape the overall network structure after the module decomposition resulted from the execution of the cluster tool. Each module is represented as a meta-node, which contains information about the inner nodes (nodes belonging to that module).

The MetaData plugin makes use of the Cytoscape MetaNode plugin to represent graphically the meta-nodes. The MetaNode plugin can be downloaded from http://labs.systemsbiology.net/galitski/ at the "Biomodules" project. It is planned for version 2.3 or 2.4 of Cytoscape to include this plugin in the main core of the software.

To use the MetaData plugin, at least 2 files should be supplied: a network file in one of the formats supported by Cytoscape and a Cytoscape node attributes file. To understand these formats, see Section 2.1.1.1.

The node attributes file should contain the information about the module separation (decomposition) of the network. Such file can be generated manually, with tools like Excel or any other software, since the file format is very simple. The Cluster tool (*libclust* – see section 2.3.2) can also generate files in this format. Figure 2.6 shows part of the regulatory network from *P. aeruginosa*, with colors representing different modules, i.e. nodes with the same color belong to the same module.

After the MetaData plugin is used, the network layout will be shown as in Figure 2.7. The nodes in the network are replaced by meta-nodes, representing the modules. The size of the meta-nodes reflects the number of nodes in that module. Bigger meta-nodes are representing more nodes and *vice-versa*.

After converting nodes into their module representation, the edges connecting the modules are still shown. This way it is possible to see how the modules are interconnected. Modules highly interconnected will have lots of edges between them.

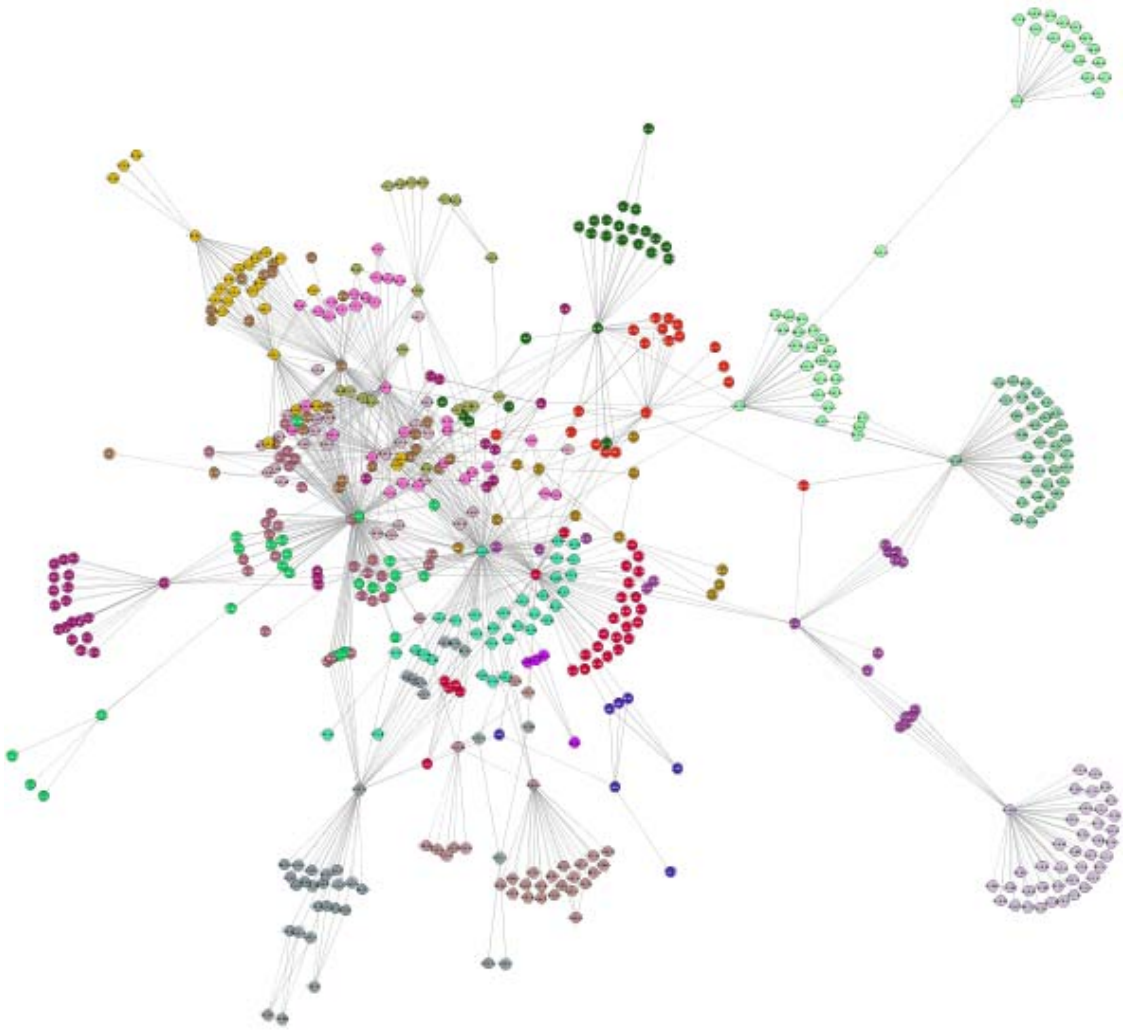


Figure 2.6 – Regulatory network from *P. aeruginosa* with different colors showing the module decomposition

It was tried to change the MetaNodes plugin to show only one node representing all the edges, with an attribute storing the number of edges the "meta-edge" would be representing. The problem is that Cytoscape does not support (yet) edges with a continuous variation in the thickness to show the number of edges being represented by the meta edges like it is done with the meta nodes (that change the size continuously to show the amount of nodes represented). Then it was decided to just show the total number of edges instead of only one edge, since it wouldn't be possible to have a good overview of module connectivity with all edges with the same thickness. An alternative

to avoid so many lines between two meta-nodes is to use only one edge together with its label indicating the number of edges it represents.

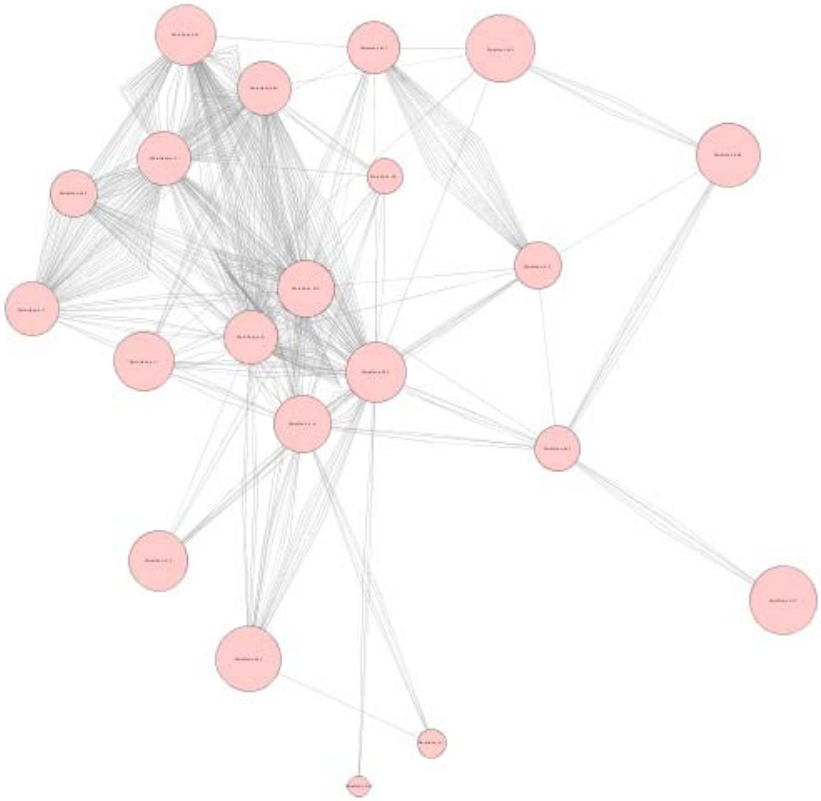


Figure 2.7 – Regulatory network from *P. aeruginosa* with nodes substituted by meta nodes representing the modules.

The MetaData plugin can also integrate experimental data showing the number of genes in each module that are up or down regulated. Figure 2.8 shows the same data from *P. aeruginosa* shown as meta-nodes with integration of data from microarray experiments.

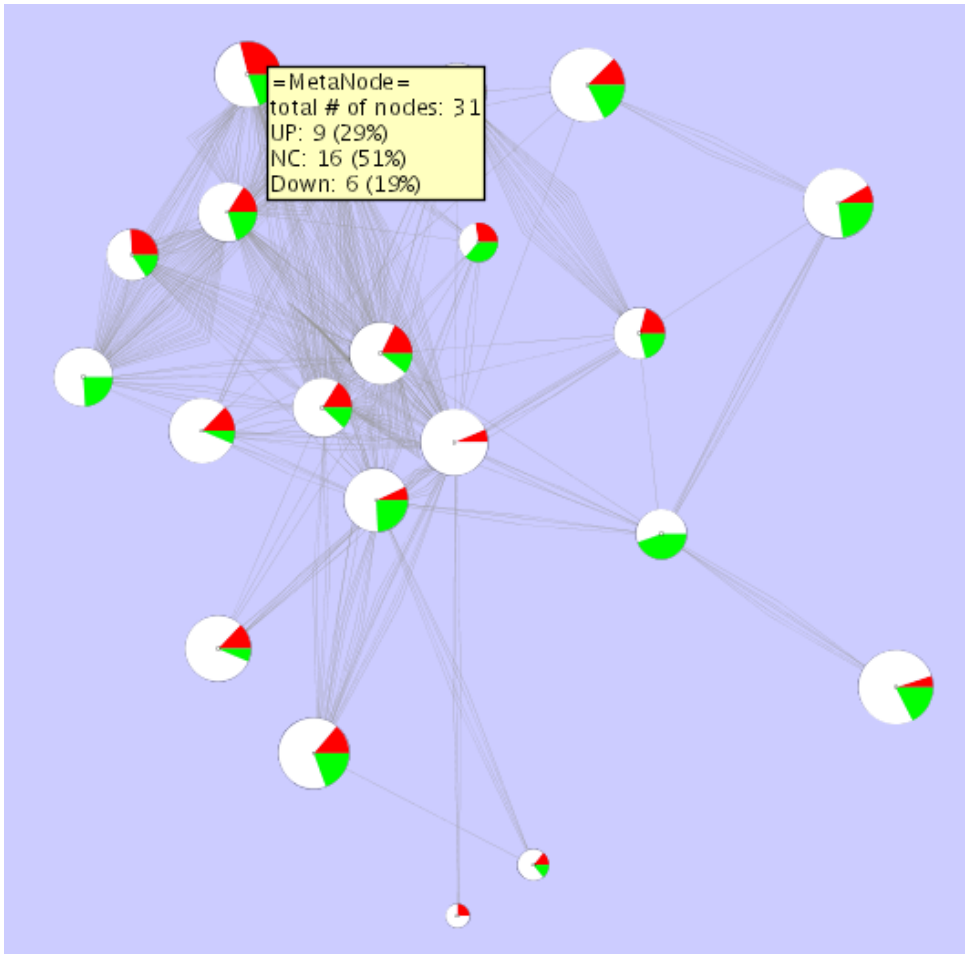


Figure 2.8 – Regulatory network from *P. aeruginosa* showing experimental data as a pie-graph.
The tooltip (in yellow) shows statistics on the expression of the genes belonging to each meta node, such as the number of nodes up and down regulated and nodes that didn't change in the two datasets compared.

Cytoscape has support for loading experimental data directly. What MetaData does is to help the visualization of such data per module instead of per node as the standard Cytoscape method does. The approach of MetaData helps to analyze the network at a systems level, instead of individually. Both methods have pros and cons, dependent on the concrete demand of the end user.

2.3 Tools for network analysis and visualization

2.3.1 Network conversion

Currently, there is a large number of software for network visualization and analysis. One of the problems working with different software is that, in many cases, they use incompatible file formats for storage of the data. As an example, in our work we often use *Pajek* (http://vlado.fmf.uni-lj.si/pub/networks/pajek/) as data analysis tool and Cytoscape for network visualization. The file format used by Pajek (.net) can not be imported by Cytoscape and *vice versa*.

Since we could not find a tool to convert between the file formats from *Pajek* and *Cytoscape*, it was necessary to create such tool. Two versions of this tool were developed.

2.3.1.1 NetConv

NetConv was the first version of the conversion tool. It was developed using the Tcl/Tk (http://tcl.tk) language. It started as a collection of scripts to do network conversion that later were joined in one tool. Figure 2.9 shows a screenshot of the application.

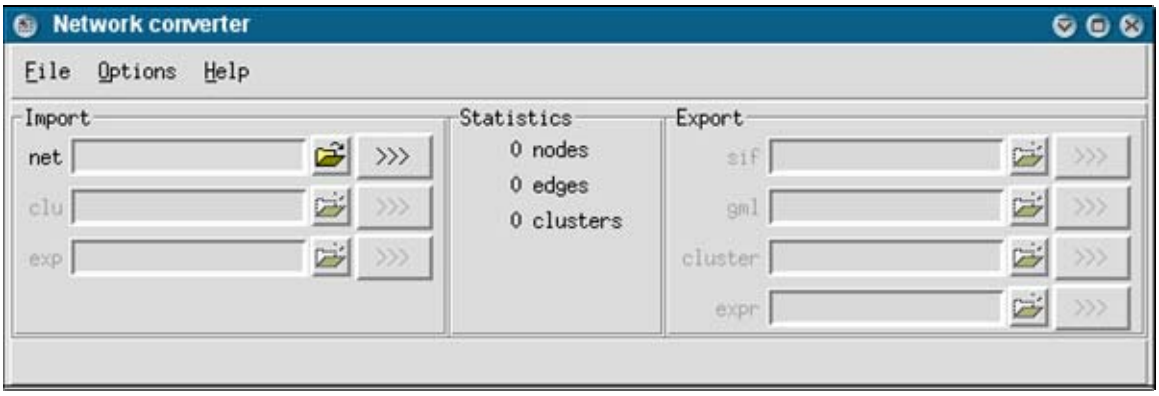


Figure 2.9 – *NetConv* version 1

The tool is able to read *Pajek* network (.net) files and *Pajek* cluster (.clu) files. It is also able to export the data to *Cytoscape* SIF format and GML. *Cytoscape* node attributes file

format is also supported for export. It was mainly used to transfer networks from Pajek to Cytoscape format as the opposite was not supported in this version.

2.3.1.2 pyNetConv

The first version of the *NetConv* tool was limited as the conversion could be made only in one direction (Pajek to Cytoscape). As that tool was not developed from the beginning to be expandable, it was not so simple to do it. Another problem with that version is that though Tcl/Tk is a great language to develop some interfaces quickly, it was not designed for speed, and it seemed not to be the best language to develop such a tool.

Then a second version of *NetConv*, called *pyNetConv*, was developed with expansibility in mind. It was developed as a library. This project is also hosted at Source Forge like the Cytoscape plugins mentioned above. See http://pynetconv.sourceforge.net .

pyNetConv was developed in Python (http://python.org) and was initially projected to be used as a library, so users can develop their own applications in Python and use *pyNetConv* to import/export files in the formats supported. *libclust* (a tool described later in this chapter) makes use of the *pyNetConv* library.

pyNetConv can import and export (read and write) the following file formats:

- Pajek networks (.net) files;
- Pajek cluster (.clu) files;
- Cytoscape networks (.sif) files;
- Cytoscape node attributes (.na) files; and
- GML files.

pyNetConv has also support to import tab-separated values created by *Affymetrix Micro Array Suite 5.0* (MAS5) and convert this data to *Cytoscape* node attributes or expression

data (.pval file). With this feature, it is possible to integrate data from microarray experiments directly into the network data and then visualize it using Cytoscape.

The main purpose of *pyNetConv* was to work as a library to enable other applications to load and save networks in the supported formats. To allow users to convert networks directly using the features of *pyNetConv* without the need to develop their own software, a graphical application that uses the library was also created and is distributed with the library. Figure 2.10 shows a screenshot of the graphic interface of this tool.

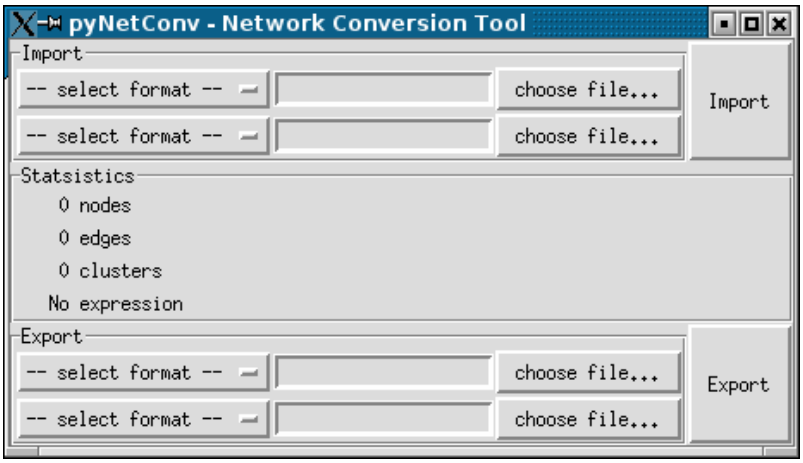


Figure 2.10 – pyNetConv graphic interface

The use of *pyNetConv* as a library is quite easy. Listing 2.3 shows an example using *pyNetConv* as a library in the development of a tool.

Listing 2.3 – Example of use of pyNetConv as a library

```
from netconv import *

n = Network()
importPajek(n, 'network.net')
importPajekClu(n, 'network.clu')
n.cluster2Attributes()
n.pajek2pos(1000)
exportCyAttributes(n, 'network.cluster')
exportGML(n, 'network.gml') # network + cluster information
```

For a reference of the available functions, read the documentation of the API of *pyNetConv* at http://pynetconv.sourceforge.net/api/ .

2.3.2 Cluster Tool

libclust is a library and a tool for clustering networks. It is developed in Python and uses *pyNetConv* to import/export networks.

First it was developed as a single application for clustering of networks, but the code was rewritten as a library to make it easier to expand the tool and also giving the user the possibility to develop his/her own application based on *libclust*.

This tool is described in details in Chapter 3 with some examples of use.

2.3.3 A tool for visualization of fermentation process

At the beginning of this work, a small tool was developed to help the visualization and simulation of the effects of metabolic overflow and grown inhibition in continuous culture (Xiu *et al.* 1998). The media containing a sole carbon source as the only one limiting factor was supplied in a continuous feeding rate. The bioreactor was considered as an ideal stirred tank. The culture produces a major product which exerts feed-back inhibition on the cellular growth. The substrate itself is also assumed to be toxic to growth at high concentration. The material balance equations of the culture can then be written as follows:

$$\text{Biomass:} \quad \frac{dX}{dt} = X\left(\mu - D\right)$$

$$\text{Substrate:} \quad \frac{dC_S}{dt} = D\left(C_{S0} - C_S\right) - Xq_S$$

$$\text{Product:} \quad \frac{dC_P}{dt} = Xq_P - DC_P$$

where X, C_{S0}, C_S, C_P are concentrations of biomass, substrate in the feeding medium, substrate in reactor, product in reactor, respectively. D, μ, q_S and q_P are dilution rate, specific growth rate, specific consumption rate of substrate and the specific formation rate of product, respectively.

In order to study the dynamic behavior of this culture system, kinetic expressions for μ, q_S and q_P are needed. The following growth model is applied:

$$\mu = \mu_{\max} \frac{C_S}{C_S + K_S}\left(1 - \frac{C_S}{C_S^*}\right)\left(1 - \frac{C_P}{C_P^*}\right)$$

q_S and q_P are defined as:

$$q_S = m_s + \frac{\mu}{Y_S^m} + \Delta q_S^m \frac{C_S}{C_S + K_S^*}$$

$$q_P = m_P + \frac{\mu}{Y_P^m} + \Delta q_P^m \frac{C_S}{C_S + K_P^*}$$

where $\mu_{\max}$ is the maximum specific growth rate; K_S is the saturation constant; C_S^* and C_S^* are the critical concentrations of the substrate and product above which cells cease to grow.

The simulation tool was implemented in Python using the graphic toolkit Tk via the Tkinter module from Python. Figure 2.11 shows screenshots of the tool running.

The simulation tool is composed by two modules: one to solve the ordinary differential equation (ODE) system and a second one to show the progress of the simulation over time in a graphical form.

(a) (b)

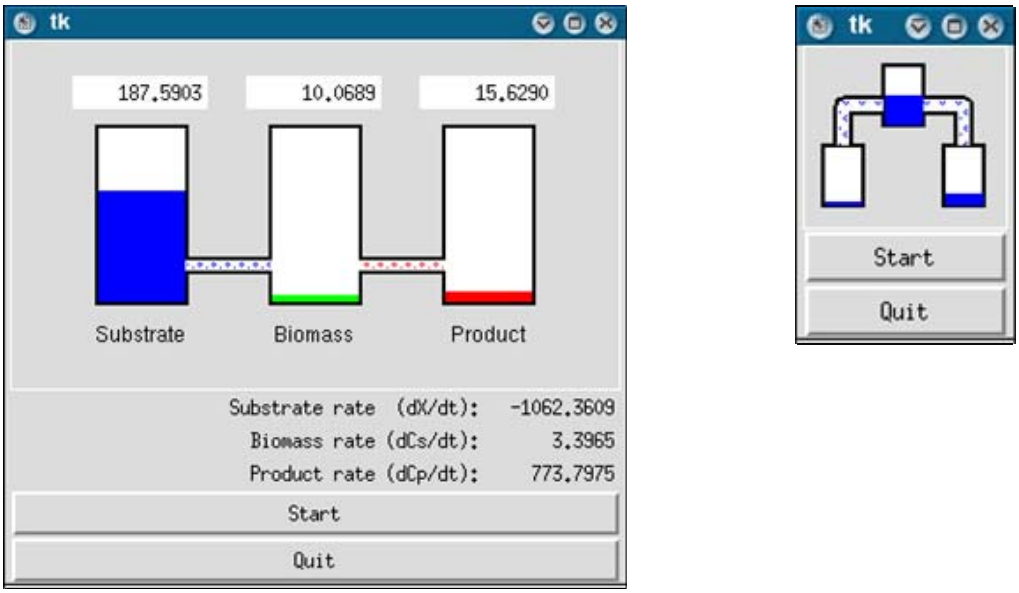


Figure 2.11 – Tool for simulating the effects of metabolic overflow and grown inhibition in continuous culture

Chapter 3

Tools for clustering

In Chapter 2, other tools developed during this work were described. Here, the *Cluster Tool* will be described in more details.

During the work with network decomposition, two versions of the cluster tool were developed. The first one is a standalone application with many options that make possible for the user to control the application.

The second version of the cluster tool was developed as a library, which makes possible for the user to develop his/her own cluster tool based on it. A standalone application that makes use of the library is included as an example. With this standalone example, it is possible for the user to try the new cluster library without the need to write any code.

In this chapter, both versions of the cluster tool are described with more emphasis on the version 2, as it is the current version of the tool. Version 1 will still be described here (and available for download), because it has some features not yet supported by version 2.

3.1 Cluster Tool, version 1

The first version of the cluster tool was developed as a standalone application. To use this tool, the following software should be installed:

- the Python (http://python.org/) language; and
- the NetworkX (http://networkx.sourceforge.net/) library.

Also, if the user wants to work with networks in other formats than a simple list of interactions (the default one supported by the cluster tool) then it is suggested to install also:

- pyNetConv (http://pynetconv.sourceforge.net/) which supports other formats like Cytoscape SIF format, Pajek NET format and GML.

The cluster tool has several options to modify its behavior. To see all available options, type:

```
$ cluster.py --help
```

in the command line. The tool will show all possible options to use. See Appendix B for a complete list of options.

To cluster a network, the user should provide the network in one of the supported formats and some parameters to the cluster tool to specify how it should be clustered.

In the following subsections an overview of the most important options that can be used with this tool are shown.

3.1.1 General options

The output of the cluster tool is saved using the Cytoscape node attributes format (for details about this file format see Section 2.1.1.1). The attribute will be the module to which each node belongs after the network decomposition finishes. The option **`-a`** is used to specify the name of the output file. When the user calls the cluster tool like this:

```
$ cluster.py -a output.na network.sif
```

The cluster tool will read the file **network.sif** in SIF format (the extension is used by the cluster tool to identify the type of file) and the output, in node attributes format will be saved to **output.na**.

It is possible to store the file in a tree format (the one used by tools like TreeView). To enable this type of output, change the **-a** option by **-t**. See Figure 4.1 for an example of output generated by the **-t** option opened in TreeView.

To make the operation of the cluster tool easier for new users, there is an option **--easy** which will enable the easy mode with the most used options turned on.

3.1.2 Selection criteria options

By default, the cluster tool version 1 will use the following criteria for the clustering:

1. **distance**: the nodes or modules with the smallest distance will be clustered first;
2. **centrality**: the nodes or modules which will generate the module with the smallest closeness centrality (module in the periphery) after clustered will be clustered first;
3. **degree**: the nodes or modules which will generate the module with the smallest total degree after clustered will be clustered first;
4. **size**: the nodes or modules which will generate the module with the smaller number of nodes after clustered will be clustered first;

If, after one criterion is applied, more then one pair of candidates for clustering exists, then the next criteria will be used following the order shown in the previous list. If at the end there is still more then one pair, then the first one will be used. This is unlikely to happen as centrality usually will give unique results. Degree and size are used only for robustness just to help the result to be unique in the (rare) case centrality will not give a unique result.

Lowest degree is used as an extra argument to let the “hubs” to be clustered later and the lowest module size is used to first generate small modules and then join them into bigger ones instead of generating big ones at the beginning.

If desired, there are options to change this behavior, eliminating some of the criteria during the clustering. It is also possible to use modularity as criterion (see Appendix B).

3.1.3 Core options

The cluster tool can be used for normal network decomposition or for decomposition based on a core-periphery structure (Holme 2005).

With these options, it is possible to control if the core-periphery should be used or not and it is also possible to set some parameters to the core creation.

3.1.4 Modules options

While decomposing a network, the cluster tool lets the user to select how the modules should be created. The user can select a maximal or minimal number of modules, a maximal or minimal number of nodes per module. It is also possible to let the tool choose the best number of modules based on modularity (see Section 1.3.12).

3.1.5 Cytoscape interaction options

It is possible for the cluster tool to communicate with Cytoscape and show, interactively, the nodes as they are clustered. For this to work, Cytoscape should have the CytoTalk plugin installed. See http://cytoscape.org/plugins2.php for info on how to get this plugin.

3.1.6 Other options

For a complete list of options supported by the cluster tool, see Appendix B.

3.2 Cluster Tool, version 2

The first version of the tool is composed by a single file with more then 3000 lines of code. After some time, the size of the file became a problem to update the tool and add new features. It was also not very flexible. To add a new feature, the code should be changed, “breaking” code that was already working and that needed to be fixed again.

Because of these problems and to add more flexibility to the tool enabling other tools to be developed using the same code base, the cluster tool was rewritten from scratch using the ideas and algorithms of the first tool.

The second version of the cluster tool was developed as a library named *libclust* and has the same requirements of the first version to be installed. It is divided in many files and each file has functions related. As it has more files then the old version, an installer tool was created to help the installation of *libclust*.

Many improvements in speed were also done in the second version of the tool. For example, to cluster the metabolic network from *E. coli*, the new version takes around 10 seconds which is much better than the first version that would take more then 10 minutes running on the same computer.

As the second version of the cluster tool was developed as a library, the user can develop his/her own application using the code-base of *libclust* and have a customized tool. As an example of utilization of the library and to be used as a standalone tool for people not wanting to write a new tool, a program called *ncluster.py* is included in the distribution of the library and it is installed automatically with the library.

The basic use of *ncluster.py* is similar to *cluster.py*. Typing:

```
$ ncluster.py --help
```

The tool will show a help explaining how to use the tool and the basic options. Listing 3.1 shows the output of this command.

Listing 3.1 – Output of ncluster.py with option --help

```
$ ncluster.py --help

usage: ncluster.py [options] network

General options:
  -h, --help            shows this help message and exit.
  -a PREFIX             sets the prefix to use in file export. Defaults to
                        base name of network.
  --core=SIZE           creates core-periphery structure instead of looking
                        for communities. Size is the initial core size. Use 0
                        to let the tool calculate it.
  --centr=C             centrality method to use with --core
                            c: closeness
                            b: betweenness
                            d: degree
  -c CRIT               sets the criteria of clustering from the pre-def. ones:
                            d: distance
                            m: modularity
                            c: centrality
                            g: degree
                            s: size
  -C CRIT               same as -c but accepts a list (one value per stage).
  -s STAGES             sets the stages from the pre-defined ones:
                            f: stop at #free nodes == 0
                            m: export modularity for each step and stop
                               at 1 module.

Extra options:
  --np                  don't show the progress bar while clustering
  -x                    export all modules composition instead of only the
                        one with best modularity
  --show-modules        while analysing the modules, show the results on screen
  --no-calc-modularity  do not calculate the modularity for every step.
  --no-save-modularity  do not save modularity at a separate file.
  --log                 save some debug info.

Advanced options:
  --config FILE         loads FILE with extra configuration.
```

To use the cluster tool in its simplest form, only the network should be informed and all the default settings will be used.

3.2.1 Basic usage

By default, the name of the output file consists of the name of the network without extension, the number of modules and the extension **`.na`** added to it. For example, for the network **`network.net`**, a file in *Pajek* format (extension **`.net`**), if the network

decomposition calculated by the cluster tool generates 10 modules, the output will be saved in Cytoscape node attributes format to the file **`network.m10.na`**.

If the user wants to have a different prefix for the output file, the option **`-a`** can be used. For example, with the option "**`-a net`**" in the example above, the output file would be **`net.m10.na`**. This is useful to run the cluster tool with different parameters and store the results in different files for later comparison.

After the clustering is finished, the cluster tool will identify the number of modules that generated the best value for modularity and will export that one to the output file.

For each step in the clustering, the nodes will be joined in modules. In the default case, using the following criteria:

1. select the nodes to be clustered using **modularity** (the modules that will give the biggest increase in modularity or smallest decrease after clustered will be clustered first);

2. if there are more then one pair of nodes to be selected, select the ones which will generate the module with the lowest **centrality** (nodes at periphery) after clustered;

3. if there are more then one pair of nodes to be selected, select the ones which will generate the module with the lowest total **degree** after clustered; and

4. if there are more then one pair of nodes to be selected, select the ones which will generate the smallest module **size** after clustered;

As for the version 1 of the cluster tool, usually only the first two criteria are used, as centrality is very likely to give a unique result. The last two criteria are added for robustness in the result.

If the user wants to have different criteria, it is enough to use the **`-c`** option, changing the order or adding other criteria like **`d`** for distance. It is possible also to define new criteria (see Section 3.2.4). The default criteria is equivalent to use "**`-c mcdg`**".

As an example, Listing 3.3 shows a network containing data from the connected part of *E. coli* metabolic network being clustered using the defaults from the cluster tool, except for the **`-a`** option. Listing 3.2 shows part of the eco-connected.sif file.

Listing 3.2 – Part of the SIF file representing the metabolic network of *E. coli*

```
527   pp    6157
178   pp    214
85    pp    1096
85    pp    644
1100  pp    1267
1100  pp    860
1168  pp    106
407   pp    3465
…
```

Listing 3.3 – Output of the cluster tool running with *E.coli* data

```
$ ncluster.py -a eco eco-connected.sif

   Network prefix: eco
     Network file: eco-connected.sif
(connected) nodes: 473
   (unique) edges: 574

Calculating distance |########################################|[100.0%]
Calculating modularity |########################################|[100.0%]
Clustering (stage1) |########################################|[100.0%]
Clustering (stage2) |########################################|[100.0%]

Best modularity: 0.846683825226 with 16 modules.

Elapsed time: 10 seconds
```

First, the output shows some information about the network to confirm the data. The number of connected nodes and unique edges is shown.

The output also shows the progress of each stage of calculation (using a progress bar), which is useful for long calculations (big networks). The number of modules with best modularity is shown and this data is exported to the output file. At the end, the time used in the decomposition is shown. Listing 3.4 shows part of the generated file **`eco.m16.na`**.

Listing 3.4 – Example of the output file generated for *E. coli* metabolic network

```
MetaData
5116 = module13
877 = module13
332 = module13
164 = module13
333 = module12
558 = module12
817 = module12
204 = module12
905 = module12
…
```

The first line shows the name of the attribute to be used in Cytoscape ("MetaData" in the example to use the results with the MetaData plugin for Cytoscape – see Section 2.2.4) and the other lines are divided in three columns: node name, a separator and the module to which the node belongs.

3.2.2 Stages

To add more flexibility, the cluster tool version 2 makes use of stages to change the behavior of the tool during the clustering.

For each stage, it is possible to use different criteria for clustering and it is possible to generate intermediate files and other information during the different stages. In the default configuration, there are 2 stages defined.

Default stage 1 will just cluster based on the defined criteria (or the default one "**`mcdg`**" if none is supplied) until there are no free nodes, that means, until all nodes were clustered at least once and belong to some module. For the default stage 2, the tool will behave similar to stage 1 except that it will save information for the modularity at each step, to choose the best one at the end.

It is possible to change the default stages using the **`-s`** option. For now, only the two default stages are defined, but more can be added to the tool in the future. It is also possible for the user to define new stages for use without changing the main code of the cluster tool as described in Section 3.2.4.2.

If needed, it is also possible to have different criteria in different stages. This is achieved using the **-C** option (note: capital "C") which is similar to the **-c** option (small caps "c") except that it accepts a list representing the criteria for each stage. For example:

```
$ ncluster.py -c dmc network.sif
```

will use **distance**, **modularity** and **centrality** (in this order) during the clustering in all stages while:

```
$ ncluster.py -C '["dc","mc"]' network.sif
```

will use first **distance** then **centrality** in the first stage and first **modularity** then **centrality** in the second stage.

3.2.3 Core-periphery decomposition

The cluster tool can be used for normal network decomposition as shown before, but it is also possible to use it for network core-periphery decomposition.

To decompose a network into core and periphery modules, the parameter **--core** should be used. The parameter passed to this option is the initial size of the core (for details on the method of core-periphery decomposition, see Chapter 5).

The syntax for decomposing a network using the core-periphery method setting the initial core size to 20 nodes is:

```
$ ncluster.py --core=20 network.sif
```

If the user wants the tool to try to calculate the initial core size, 0 (zero) should be used as the initial core size:

```
$ ncluster.py --core=0 network.sif
```

The default method for choosing the nodes to form the initial core is closeness centrality. To change this behavior, use option **--centr**. For more details on core extraction see Section 5.4.

3.2.4 Advanced usage

The basic usage of the cluster tool is enough for most of the cases, but sometimes it is needed to customize the tool to create a new algorithm or test different forms of decomposition. This section describes the methods to do this customization and tips to use the tool for working with many networks.

3.2.4.1 Batch clustering

It is possible to run the cluster tool in batch jobs, allowing more then one network to be clustered in sequence (or in parallel if desired). This can be done, for example, in a dedicated computer that stays on 24h per day. In such cases, only a shell access is needed as the cluster tool doesn't require a graphic interface to run.

When running the cluster tool in a batch job or when clustering a big network, it may be desired to save the output of the tool (the one that goes to screen, not the node attributes file) to a separated file to check later, as the tool may be running overnight.

In these cases, the option **`--np`** can be used to suppress the output of the progress bars, to make the output file smaller.

3.2.4.2 Extending the Cluster Tool

The cluster tool is very flexible and it can be customized without the need to change the main code. For the tool customization, knowledge of the Python language is required.

To customize the tool, the **`--config`** option is used. With this option, it is possible to extend the tool without the need to create a new application or modifying the existing code.

To make use of this option, the user should create an extra file with code defining new selection criteria and/or stages that can be used by the *ncluster.py* tool instead of the built in ones.

After creating the extra file with the new definitions, just run the tool like this:

```
$ ncluster.py --config defs.py network.sif
```

and the new definitions will be used in the clustering.

The code in the **defs.py** file is loaded after all other functions are loaded and before the cluster starts. Because of this, it is possible to redefine (by overriding) any part of the tool by adding instructions in this file. Table 3.1 shows some useful variables that can be redefined and some useful functions to use in the customization file.

Table 3.1 – Special variables and functions for cluster tool customization

Variable	Description
criteria	list containing the functions to be used as criteria
stages	list containing the definitions of the stages
getDefaultCriteria()	function that returns a list with the default criteria — useful for small modifications
getDefaultStages()	function that returns a list with the default stages — useful for small modifications

getDefaultCriteria() and **getDefaultStages()** are useful in the case the user wants to do a small modification based on the original settings. In this case, it is not needed to define all criteria and stages to be used; only the new ones need to be defined and the tool will use the default ones for the other parts. See example in Listing 3.5.

The format of the **criteria** list is just a list with the functions to be used as criteria in the order they should be used.

For the stages list, a more elaborate format is defined. It is actually a list of lists, and each internal list describes one stage, based on the following format:

```
[ f_step pb_label pb_max pb_step ]
```

where:

f_step: function to be executed in each step of the stage. Should return **True** at end of stage, **False** otherwise.

pb_label: label to be used in the progress bar.

pb_max: function that returns the max value for the progress bar.

pb_step: function that returns the value to set progress bar in each step.

Note that the customization can be done by changing any part of the tool, not only the special variables. It is possible to modify any part of the code, redefine functions etc. The use of the special variables is recommended because it is the standard way to modify the tool, but the user is not limited to them.

For example, Listing 3.5 shows an example of **config** file.

Listing 3.5 – Example of config file

```
def newCriterion(G):
    # define new criterion here

def stopAt20FreNodes(G):
    return len(G.freeNodes) == 20

def testEndBar(G):
    return G.nnodes-20

def stepBar(G):
    st = len(G.done)-20
    if st < 0:
        return 0
    return st

criteria = getDefaultCriteria(G)
criteria.insert(0, newCriterion)

stages = getDefaultStages(G)
stage[0] = [stopAt20FreNodes, "new criterion",
            testEndBar, stepBar]
```

The code in Listing 3.5 modifies both, the default criteria and the default stages.

For the default criteria, the new defined criteria (function **newCriterion()**) is defined and inserted at the beginning of the criteria list (position 0) by the line:

```
criteria.insert(0, newCriterion)
```

That means, this new criterion will be used first and then the default ones will follow.

For the stages modification in the sample code, a new stage was defined to substitute the first default stage (note that lists index numbering in Python, as in C, starts from zero). The second stage will be used as defined by the default function.

For this new stage, three functions are defined, one to test the end of the stage and other two for the update of the progress bar. These two functions may be set to **`None`** if the progress bar is not used (option **`--np`** of *ncluster.py*).

There are some examples of config files included in the *libclust* distribution in the **`samples/`** folder.

3.2.5 Example of use

For this example, we create a small file in SIF format to be clustered. Listing 3.6 shows this file and Figure 3.1 shows it visualized in Cytoscape.

Listing 3.7 shows the output of the cluster tool running with this file as input. After this, the file **`tt.n3.na`** is generated. Its contents are shown in Listing 3.8. Loading the file **`tt.n3.na`** in Cytoscape with the network file **`test.sif`**, the modules are shown in different shapes and colors as shown in Figure 3.2.

If desired, the modules can be shown as meta-nodes using the MetaData plugin (see Section 2.2.4) as shown in Figure 3.3 instead of being represented through different colors.

For other examples of use of the cluster tool, see chapters 4 and 5.

Listing 3.6 – SIF file representing the sample network

```
a pp b
b pp c
c pp a
d pp e
e pp f
f pp d
g pp i
i pp h
h pp g
j pp l
j pp h
l pp k
k pp j
c pp j
l pp f
k pp g
```

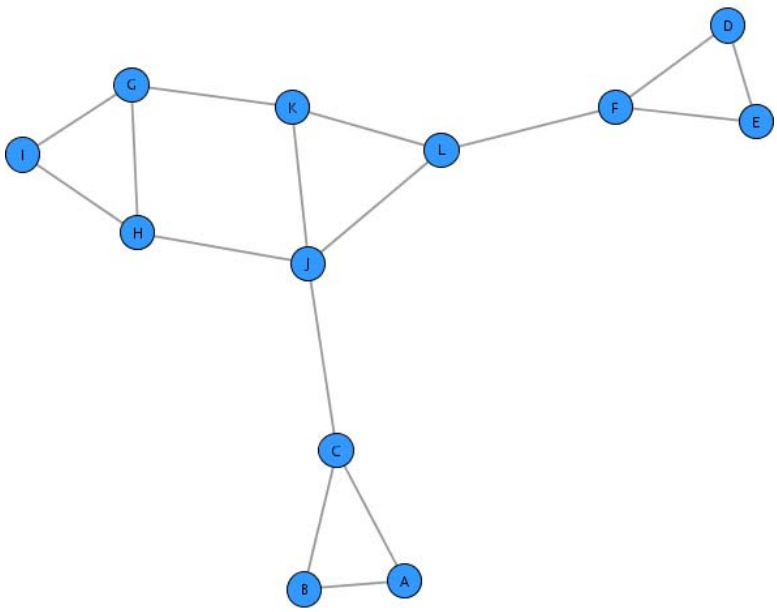


Figure 3.1 – Sample network visualized in Cytoscape

Listing 3.7 – Output of cluster tool for sample file

```
$ ncluster.py -a tt teste.sif

   Network prefix: tt
     Network file: teste.sif
(connected) nodes: 12
   (unique) edges: 16

Calculating distance |########################################|[100.0%]
Calculating modularity |########################################|[100.0%]
Clustering (stage1) |########################################|[100.0%]
Clustering (stage2) |########################################|[100.0%]

Best modularity: 0.462890625 with 3 modules.

Elapsed time: 0 seconds
```

Listing 3.8 – Output of clustering for file *test.sif*

```
MetaData
j = module3
k = module3
l = module3
h = module3
g = module3
i = module3
c = module2
a = module2
b = module2
f = module1
e = module1
d = module1
```

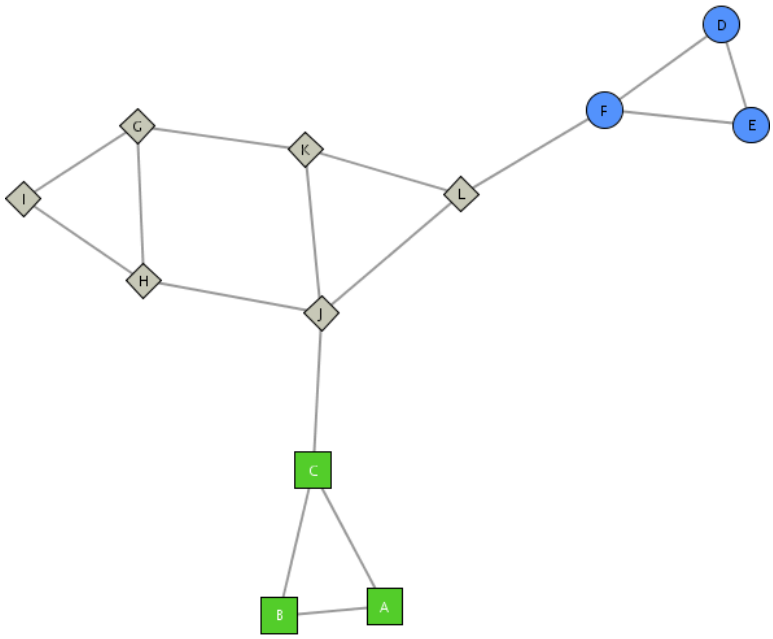


Figure 3.2 – File test.sif shown in Cytoscape with the decomposition information shown in colors

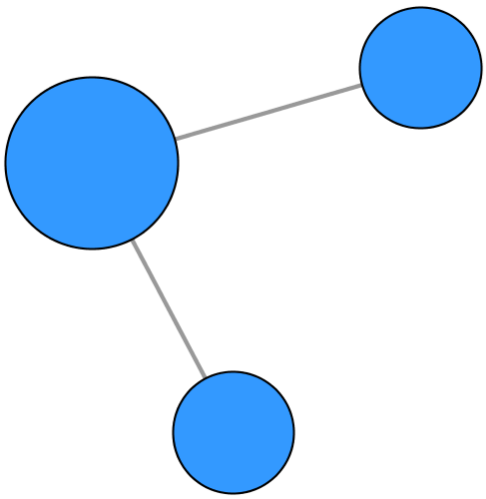


Figure 3.3 – File test.sif shown using meta nodes to represent the modules.

3.2.6 Tools included in the package

The new version of the cluster tool is distributed as a package including the library, a clustering application to use the library (*nclusterl.py*) and some other applications based on *libclust* that are useful for further analysis of the networks. In Table 3.2 these tools are described.

Table 3.2 – Tools distributed with *libclust*.

Tool	Description
checkCore.py	given a network and the results of a decomposition, this tool will search for the core module, giving also a parameter of certainty. See Section 5.2.1
metanode.py	this is an alternative to the MetaData plugin (see Section 2.2.4). It generates a network based on the decomposition with nodes representing the modules.
verifyCoreCoefficient.py	calculates the core coefficient for a given network. See Section 5.2.2.1.
networkProperties.py	displays basic analysis of a network showing the most common parameters.
compareCentrality.py	creates a tab-separated file with the calculations of betweenness, closeness and degree centralities for each node in the network showing also the metabolites in the central metabolism.
fragilityVsCentralities.py	creates a table with the nodes ordered by fragility and shows also the values for degree, closeness and betweenness centralities for each node. See Section 6.1.
orderNodesByFragility.py	analyzes the network, calculates the fragility for all nodes and generates a list of nodes ordered by fragility. Used to generate file for *fragilityVsCentralities.py*.
checkModules.py	checks the modules comparing with the KEGG database (Kanehisa & Goto 2000) showing the pathways present in each module.
libclust/keggcompound.py	library add-on. Used to interact with the KEGG database.
libclust/metnet.py	library add-on. Used to get the metabolite name based on the compound number. Faster then *libclust/keggcompound* but more limited.

Chapter 4

Network Decomposition

Working with complex networks can be very difficult if one tries to analyze the whole network at once. While working with networks with thousands of nodes it can be very hard to analyze each node and link individually and their interdependence as a whole. Also it is not possible apply analysis techniques like elementary flux modes or extreme pathways (Schilling *et al.* 2000; Schilling & Palsson 2000; Schuster *et al.* 1999; Schuster *et al.* 2000) to large scale networks. To facilitate these analyses and help to identify the most important parts of a network one may need to decompose the network into smaller parts. It is possible to decompose a network based on shortest path length, modularity, different types of centrality, etc. In this chapter, some of these techniques are presented.

4.1 Introduction

Networks are present on many systems, where sets of vertices are joined together in pairs by edges (Strogatz 2001). Examples include social networks (Amaral *et al.* 2000; Newman 2001; Scott 2000; Wasserman & Faust 1994; Watts & Strogatz 1998),

technological networks such as the internet, the World Wide Web (Albert *et al.* 2000) and biological networks such as neural networks, protein interaction networks and metabolic networks (Ma & Zeng 2003; Jeong *et al.* 2000; Guimera & Nunes Amaral 2005).

Recent research on networks among mathematicians and physicists has focused on a number of distinctive statistical properties that most networks seem to share. One such property is the "small world effect", which means that any two nodes in the system can be connected by relatively short paths along existing links. Another is the right-skewed degree distributions that many networks possess. There are typically many vertices in a network with low degree and a small number with high degree, the precise distribution often following a power-law or exponential form (Amaral *et al.* 2000; Barabasi & Albert 1999; Ma & Zeng 2003; Strogatz 2001).

A third property appears to be common to many networks, the presence of a *community structure* (Girvan & Newman 2002), which is also sometimes called clustering. Considering the case of social networks (networks of friendships or other acquaintances between individuals), it is a matter of common experience that such networks seem to have communities in them: subsets of vertices with dense node-node connections, but with less dense connections between the subsets. It is also possible that the communities join together to form meta-communities, and those meta-communities are joined together, and so on in a hierarchical fashion (Ravasz *et al.* 2002; Ravasz & Barabasi 2003).

4.2 Methods for network decomposition

There are many approaches to network decomposition. Some of them try to analyze the network as a whole and identify the communities (Kirkpatrick *et al.* 1983; Comellas & Miralles 2005) while others try to work locally in the network (Bagrow & Bollt 2005). In this work networks were decomposed using hierarchical clustering. To achieve this, one first calculates a weight W_{ij} for every pair i, j of vertices in the network, which represents in some sense how closely connected the nodes are. Then the n vertices in the network are taken, with no edges between them, and edges are added between pairs one by one in order of their weights and progressing to the weakest. As edges are added, the

resulting graph shows a nested set of increasingly large components (connected subset of vertices), which are taken to be the communities. Since the components are properly nested, they can all be represented using a tree of the type shown in Figure 4.1, in which the lowest level at which two vertices are connected represents the strength of the edge which resulted in their first becoming members of the same community. A "slice" through this tree at any level gives the communities which existed just before an edge of the corresponding weight was added. Trees of this type are sometimes called *dendrograms* in the social network literature.

Many different weights have been proposed for use with hierarchical clustering algorithms. One possible definition of the weight is the number of node-independent paths between vertices. Two paths which connect the same pair of vertices are said to be node-independent if they share none of the same vertices other than their initial and final vertices (Girvan & Newman 2002).

Other common criteria to decide which nodes will be joined as a module at each stage of the decomposition is *distance*. To decompose a network based on distance (Ma *et al.* 2004), the first step is to give a proper distance definition to show the dissimilarity between the nodes. With the distances calculated one can then build a hierarchical tree for finding clusters of functionally closely related reactions. An example of such tree is showed in Figure 4.1 for the decomposition of *E. coli* network using distance.

Different types of centrality (betweenness centrality, closeness centrality, and degree centrality – see Section 1.3.9 for the definition of these centralities) can also be used to cluster (decompose) the network. Centrality cannot be used as edge weight like in the distance-based clustering because it dos not apply to edges but to nodes. It is used to classify the nodes directly. Centrality can be used as the primary criterion for clustering (Comellas & Miralles 2005) or as a complementary criterion (see Section 4.3.1).

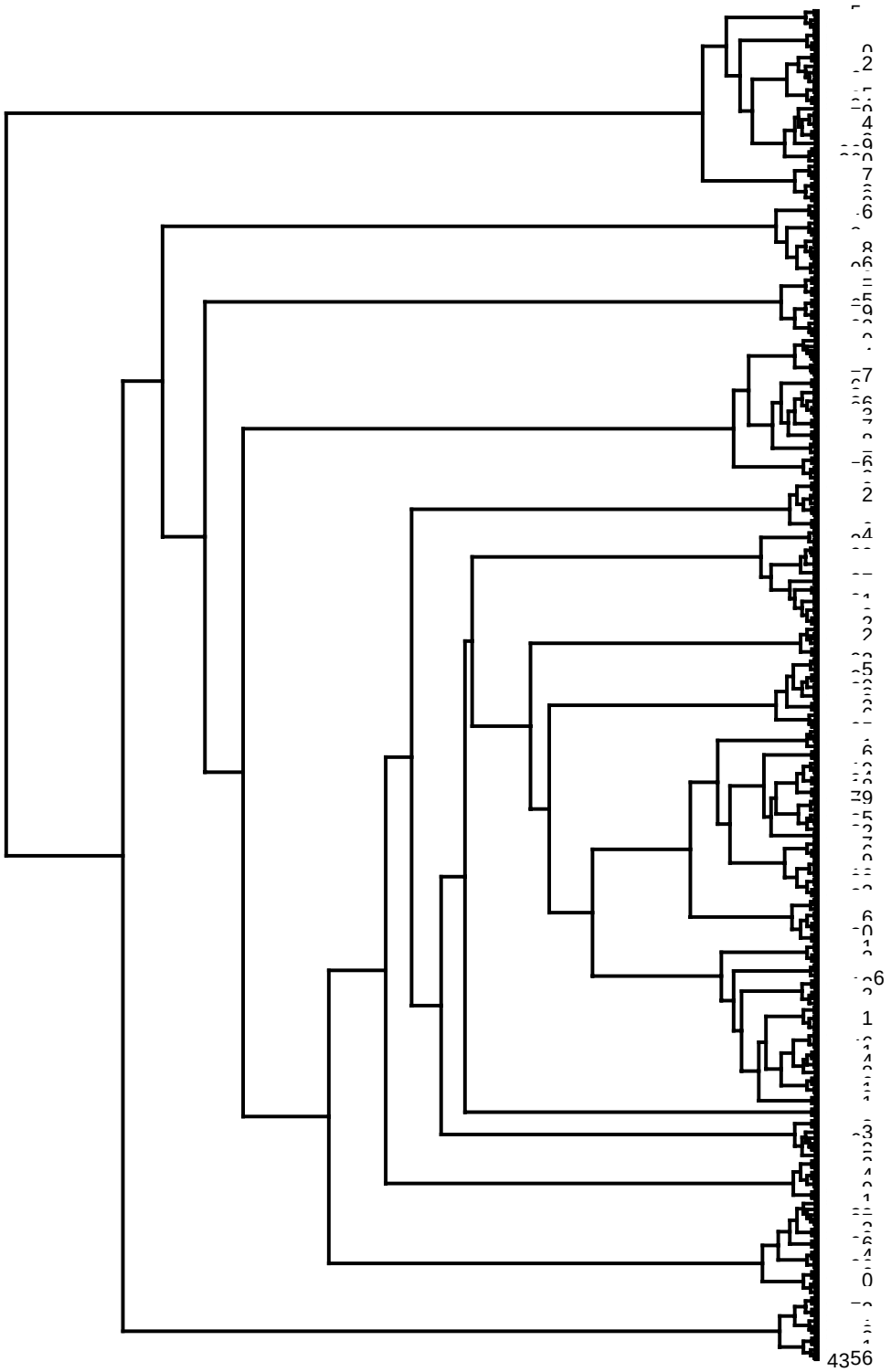


Figure 4.1 – Dendrogram showing the decomposition of *E. coli* metabolic network based on distance of nodes.

4.3 Use of modularity for network decomposition

Newman proposed (Newman 2004) to use modularity not only as a measure of the quality of network decomposition (see Section 1.3.12), but also as a parameter for the clustering.

The objective is to optimize modularity over all possible divisions to find the best one. The problem is that the optimization of modularity is very costly. According to Newman, it should be infeasible for systems larger than 20 or 30 vertices. Although various approximate optimization methods are available: simulated annealing (Kirkpatrick *et al.* 1983), genetic algorithms, and so forth. As an alternative, a "greedy" optimization algorithm is used where nodes are clustered in successive steps (Newman 2004).

In this algorithm, the network is considered as each node is a module itself, so at the beginning, the number of modules is the same as the number of nodes. Then, the modules are joined in pairs forming new modules, choosing at each step the join that results in the greatest increase (or smallest decrease) in modularity. The progress of the algorithm can be represented as a "dendrogram", a tree that shows the order of the joins (Figure 4.1 shows an example of decomposition of *E.coli* metabolic network data). Cuts through this dendrogram at different levels give divisions of the network into larger or smaller number of modules and the best cut can be chosen by using the maximal value of modularity as a criterion.

To use modularity as a criterion for clustering, it is possible to calculate the change in modularity caused by the union of modules i and j based on equation 1.6 (Newman & Girvan 2004). The change in modularity can be calculated as:

$$\Delta M_{i,j} = \frac{l_{i,j}}{L} - \left(\frac{d_{i,j}}{2L} \right)^2 \tag{4.1}$$

where $l_{i,j}$ is the number of links between the two modules, L is the total number of links in the network and $d_{i,j}$ is the sum of the degree of the nodes in the two modules.

To decide where to cut the dendrogram, modularity coefficient is used (see Section 1.3.12). The cut is done at the point that will generate a module distribution with the highest value of modularity.

The cluster tool *libclust* has an implementation of this algorithm. It was used to decompose real data (Ma & Zeng 2003) and compare with the results from Guimerà (Guimera & Nunes Amaral 2005) that used a simulated annealing algorithm to achieve similar results. Guimerà also used modularity at the end to choose the best decomposition. Table 4.1 shows a comparison of these results.

Table 4.1 – Comparison of simulated annealing and modularity methods for network decomposition

Organism	Simulated Anneling		Modularity		
	# modules	modularity	# modules	modularity	Change
A. pernix	12	0.796737	12	0.792311	-0.56%
B. subtilis	17	0.850652	16	0.845426	-0.61%
E. coli	17	0.851355	14	0.842781	-1.01%
S. cerevisiae	16	0.842338	15	0.837214	-0.61%
H. sapiens	18	0.869621	18	0.865973	-0.42%

From the results we see that the modularity-based method presented slightly lower values of modularity compared to the simulated annealing method. This difference is around 1% or less. However, the modularity method is much simpler (and probably faster) for calculation. Also the number of modules decomposed is similar in both methods. This shows the similarity in the results from both methods as the same parameter is used at the end to choose the best module decomposition.

This raises a question: how can the simulated annealing method give a better result then the one based on modularity? If modularity is the measure used at the end to test the quality of the result, how another method can present better results then the one that create communities by selecting modules that will increase more (or decrease less) the modularity?

The answer to this question is that this happens because the modularity-based method analyzes the modules to be clustered isolated from the network, while the simulated annealing method will analyze the whole network at once. That means, as the modularity-

based method will always try to increase the modularity, it may reach a local maximum. Figure 4.2 shows a diagram that illustrates how this happens.

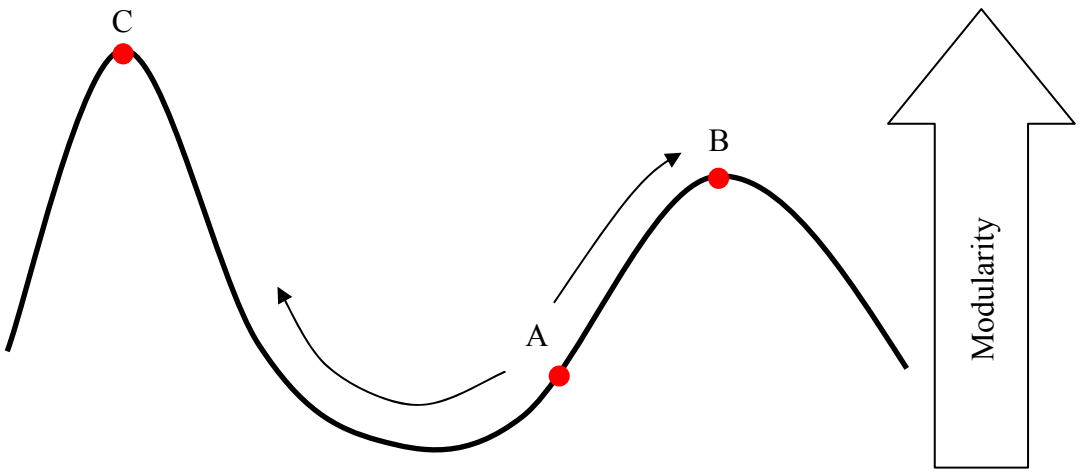


Figure 4.2 – Local maxima found by modularity calculation

Let's consider point **A** in Figure 4.2 as the initial value for modularity of the network. In the picture the modularity increases as indicated. The best modularity value that can be reached in this network is at point **C**, but to reach this point from point **A**, the value of modularity should decrease in the first steps of the clustering, but the algorithm doesn't allows this. The algorithm will always try to maximize the modularity at each step. Because of this, the algorithm will probably reach point **B**, which is a local maximum. This may happen at each stage of clustering; sometimes it is necessary to decrease the contribution of modularity to arrange the modules in a way modularity will increase much more in later stages. The only way to avoid this is by analyzing all possible combinations of nodes to form the modules in a way the modularity will be maximized at the end.

Algorithms to consider all the possible ways to cluster a network will take a very long time to finish and depending on the number of nodes in the network they may never

finish. According to Newman (Newman 2004), this kind of algorithms are only applicable to networks up to 30 nodes.

Algorithms like simulated annealing, try to make an analysis of the full network at once, but the method doesn't try all possibilities; instead it uses probability to shorten the time what reduces the robustness of the algorithm.

4.3.1 Improvement of robustness of the modularity method

The simulated annealing method is based on probability. This means that, running the algorithm many times, different results will be generated. While all results are acceptable as Guimerà shows in his work (Guimera & Nunes Amaral 2005), the method is not really robust. With the modularity decomposition method, this doesn't happen.

Using only modularity as criterion to choose the candidates for clustering may be sometimes also not very robust. This is because, especially at the beginning of the clustering, many pairs of nodes may contribute the same for the modularity (same ΔM) and, depending on which pair is chosen, the final results may be different. If in the case of more then one pair of candidates for clustering exists, this problem can be minimized by choosing always the first pair. This way we can run the algorithm many times and it may give the same result.

But how is the list of candidates generated? In which order should the candidates appear in this list? Probably based on the order they are stored in memory, which can be based in the order the nodes are read from a file. In this case, changing the order of the nodes in the input file can lead to different results, decreasing the robustness of the algorithm. This may be a problem if the network is automatically generated from software.

To avoid this problem, a multi-criteria approach is used in the cluster tool *libclust*. For each step in the calculation, the following criteria are used:

1. first, select to cluster the modules which will increase more (or decrease less) the final modularity of the network, based on equation 4.1;

2. if more then one pair of nodes exist that fits on the first criterion, then select the ones with lower closeness centrality (modules at the periphery);

3. if there are still more then one pair of candidates for clustering, choose the one that will generate a new module with the smallest total degree;

4. if the last criterion selects more then one pair, choose the one which generates a new module with the smallest number of nodes;

5. and finally, if none of the above criteria select a single pair of nodes, select the first one from the generated list.

Criteria 2-4 are added for robustness. Having more then one criteria will increase the chances of having a unique result. This way, it doesn't matter how many times the algorithm is run neither the order the nodes are stored in memory.

Criterion 1, modularity, is the same as used in Newman's algorithm (based on Equation 4.1). The second criteria, centrality, will cluster nodes on the periphery first. This will make the clustering run from the periphery to the center. This criterion will mostly give a unique result as the list of candidates is already short because of the selection based on the first criterion. This was tested − running the software step by step (debug mode) − and it is the case most of the time during the decomposition in the networks used. In the rare case it doesn't give a unique result, the other 2 criteria are used. Both will contribute to the creation of small modules first, which helps the algorithm to run more smoothly.

The results for the multi-criteria method are similar to the one using only modularity showing, in most cases, a slightly improvement in the final value of modularity as shown in Table 4.2. Table 4.3 shows this method compared to the simulated annealing method.

Table 4.2 – Comparison of modularity and multi-criteria methods for network decomposition

Organism	Modularity		Multi-criteria		
	# modules	modularity	# modules	modularity	Change
A. pernix	12	0.792311	12	0.792013	-0.04%
B. subtilis	16	0.845426	17	0.846591	0.14%
E. coli	14	0.842781	16	0.846684	0.46%
S. cerevisiae	15	0.837214	15	0.838768	0.19%
H. sapiens	18	0.865973	17	0.866001	0.00%

Table 4.3 – Comparison of simulated annealing and multi-criteria methods for network decomposition

Organism	Simulated Anneling		Multi-criteria		
	# modules	modularity	# modules	modularity	Change
A. pernix	12	0.796737	12	0.792013	-0.59%
B. subtilis	17	0.850652	17	0.846591	-0.48%
E. coli	17	0.851355	16	0.846684	-0.55%
S. cerevisiae	16	0.842338	15	0.838768	-0.42%
H. sapiens	18	0.869621	17	0.866001	-0.42%

4.3.2 Improvement of the speed of algorithm

While the method based on modularity is considered a fast algorithm (Newman 2004), its speed can be improved.

The value for ΔM can be calculated at a constant time, but it needs to be calculated for all possible combinations of modules that can be joined to choose the pair that will give a better modularity after clustered. This can take a long time for big networks.

An alternative approach, used by our cluster tool (Chapter 3), is to store the value for modularity in a table and, after each step of clustering, update this table. This will make more use of memory and may create problems with huge networks (that use lots of memory) but, as the difference in time of calculation shows, it may be unfeasible to wait for the calculation of such huge networks without the optimization of the speed anyway.

This method will start like the normal one, calculating modularity for all possible combinations of modules and choosing the first pair to cluster. The difference is that all this calculated values are stored in a table and, after the selected modules are joined, only the modules connecting to the two joined modules are recalculated and the table is updated.

Doing the clustering this way, the time to recalculate all the possible contributions to modularity at each step of clustering is reduced to a lookup in the table to find the candidates with a larger contribution to modularity.

This same approach is used for all criteria that can be used in the clustering like distance and centrality for example. Also the connectivity of the network is tracked in a similar way, updating the connectivity table after each step of clustering.

By using this approach it was noticed an increase in speed up of more then 20x depending on the size of the network.

4.3.3 Results of decomposition

In this section, results of network decomposition using the modularity method (actually the multi-criteria version) are shown for the metabolic networks studied (see Section 1.2.2).

For testing the algorithm with real-data, the metabolic networks of 5 organisms representing different families were used: *A. pernix* (archea), *B. subtilis* (gram positive bacteria), *E. coli* (gram negative bacteria), *S. cerevisiae* (eukaryote) and *H. sapiens* (more complex eukaryote).

The results are summarized in Tables 4.2 and 4.3 (multi-criteria method) for all organisms. To make this text shorter, only the data from *E. coli* will be shown in more details but the data for other organisms is also available (see Appendix D).

Using the *E. coli* data as an example, we can see in Figure 4.3 the module decomposition generated by the algorithm. In this picture, nodes represent modules and the node size is

proportional to the number of metabolites in the module. Edge labels show the number of connections between the modules. Edge labels were used to represent the number of connections instead of drawing all the edges to make the picture clearer.

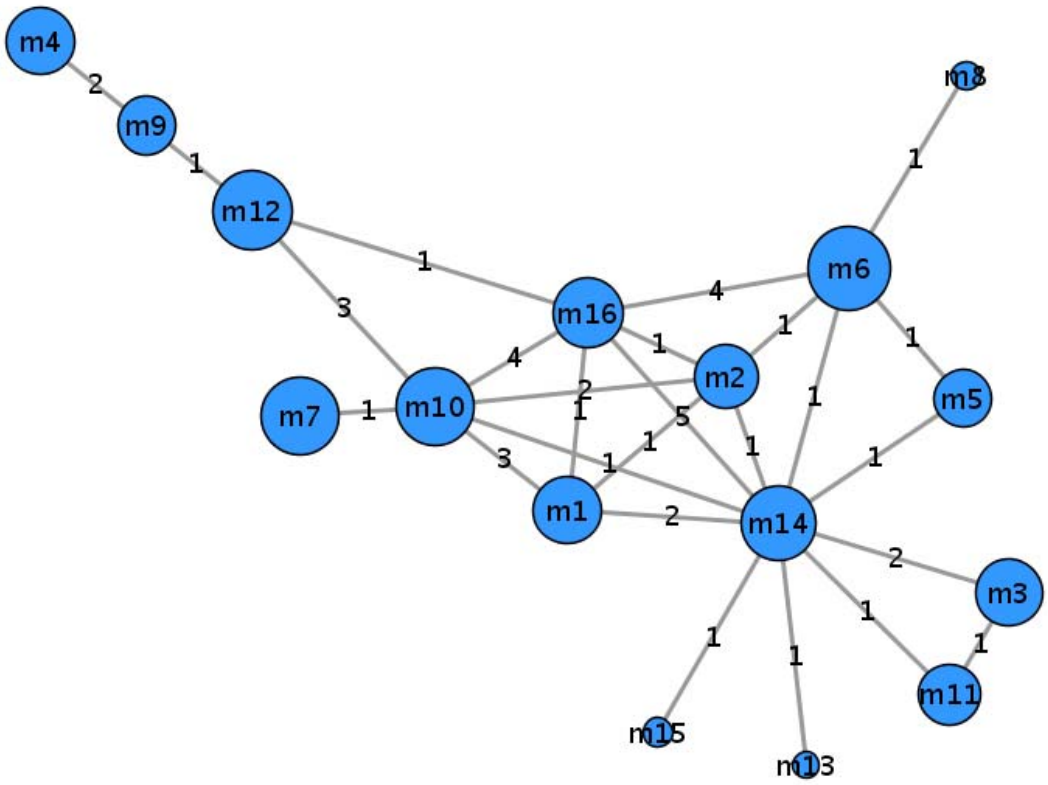


Figure 4.3 – Module decomposition for *E. coli* based on modularity algorithm

To check the results, let us consider the module 11 (marked at Figure 4.3 as **`m11`**). The nodes in this module are shown in Table 4.4 with their KEGG compound numbers. For a complete list of modules, see Appendix D.

A search in the KEGG database shows that all these nodes belong to *pyrimidine* metabolistm (KEGG http://www.genome.jp/kegg/pathway/map/map00240.html). Figure 4.4 shows these nodes highlighted in the corresponding KEGG map.

Table 4.4 – Metabolites in module 11 from *E.coli* decomposition

C00015	UDP
C00055	CMP
C00063	CTP
C00075	UTP
C00105	UMP
C00106	Uracil
C00112	CDP
C00178	Thymine
C00214	Thymidine
C00239	dCMP
C00295	ORO
C00299	Uridine
C00337	DHORO
C00363	dTDP
C00364	dTMP
C00365	dUMP
C00380	Cytosine
C00438	CAASP
C00458	dCTP
C00459	dTTP
C00460	dUTP
C00475	Cytidine
C00526	Deoxyuridine
C00705	dCDP
C00881	Deoxycytidine
C01103	OMP
C01168	Pseudouridine 5'-phosphate
C01346	dUDP
C02376	5-Methylcytosine

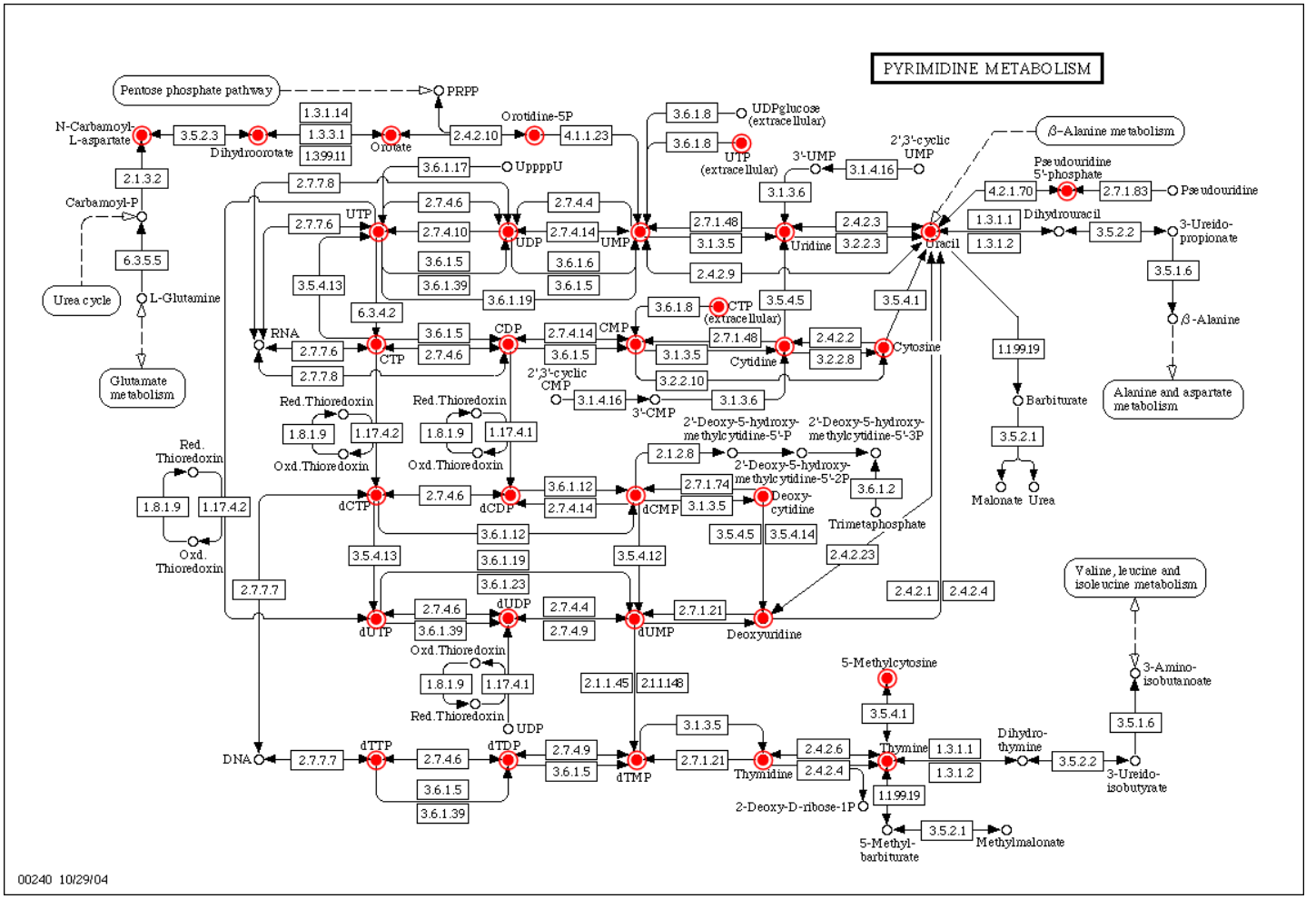


Figure 4.4 – KEGG map of Pyrimidine metabolism with metabolites in module 11 from *E. coli* decomposition highlighted

Another example that can be mentioned is that all metabolites from the TCA cycle are arranged in module 14. Further analysis of the results shows similar results for other known pathways. Table 4.5 shows the pathways present in the modules resulting of the decomposition.

Table 4.5 – Pathways in the different modules obtained by the decomposition method proposed based on modularity for *E. coli*.

Module	Pathways
13	Benzoate degradation via CoA ligation Butanoate metabolism
12	Pentose and glucuronate interconversions Purine metabolism Pentose phosphate pathway
11	Pyrimidine metabolism
10	Fructose and mannose metabolism Pentose and glucuronate interconversions
15	Valine, leucine and isoleucine degradation
9	Purine metabolism
8	Pantothenate and CoA biosynthesis
7	Galactose metabolism Nucleotide sugars metabolism
6	Glycine, serine and threonine metabolism Methionine metabolism
5	Lysine biosynthesis Peptidoglycan biosynthesis
4	Folate biosynthesis Riboflavin metabolism Purine metabolism
3	Arginine and proline metabolism Urea cycle and metabolism of amino groups
2	Phenylalanine, tyrosine and tryptophan biosynthesis Glycolysis / Gluconeogenesis
1	Glycerophospholipid metabolism Glycerolipid metabolism
14	Citrate cycle (TCA cycle) Reductive carboxylate cycle (CO2 fixation)
16	Valine, leucine and isoleucine biosynthesis

From Table 4.5 we can see that the decomposition using the modularity method does a good job in the separation of the metabolites in modules with similar functionality. Some metabolites take part in many different parts of the metabolism but in the network there is only one node representing this metabolite. This may cause parts of the network to be not well classified as these nodes need to be in classified in one module and each of them can

be in only one module. The algorithm will choose to put each of these nodes in the module that will generate the best community structure. One way to try to avoid this problem is by the use of reaction networks, where nodes represent reactions instead of the normal representation with nodes representing metabolites (Ma *et al.* 2004).

Chapter 5
Core-periphery structure in networks

In this chapter methods to decompose a network into a core-periphery structure are proposed. Also ways of detecting such structure in networks are presented.

5.1 Introduction

Many biologic networks have a core-periphery structure (Holme 2005). In such networks, not only the nodes are organized into communities as shown in Chapter 4, but these communities have different roles and organization.

In a network organized in a core-periphery structure, the core module (or community) has the function of communicate the periphery modules, so its components have relation with many modules.

After decomposing a network into communities, if the core-periphery structure is present, it is useful to identify which one of the modules play the role as the core module.

5.1.1 Core Definition

In a network organized in a core-periphery structure, one module plays the role as the "connector" from the periphery modules. This module should be connected to most (if not all) the periphery modules and should be the main way of these modules to communicate with other modules.

So, the core is composed by central nodes that have connections to different modules or, in the case of metabolic networks, to different parts of the metabolism, working as a bridge and supplying metabolites to those different pathways.

5.2 Core Detection

Not all networks present a core-periphery structure. For networks presenting such structure, there will be a module, usually a central one, which will be working as a link to different parts of the network. Here two methods to detect such structure are proposed: after clustering the network and searching for the module that fits better into the characteristics of a core module; and without clustering, using the concept of capacity of network.

5.2.1 After clustering

After clustering, if the network has a core-periphery structure, it should be possible to identify the core by inspection of the generated modules.

To find the core, one should search for a module which has the following characteristics:

1. it should have a high connectivity as the core should interconnect other modules;
2. it should have a high number of external connections (external degree) as many pathways will go trough the core;

3. it should be a central module (when looking at modules connectivity), but not necessarily the most central one;

4. its nodes should belong to the most central ones (not necessarily all of them); and

5. it may present the higher value for external degree / internal degree ratio.

Note that not all of these items *should* be present. The more of these criteria is matched, the better. If more of the above characteristics is found to be true in the network then it is higher the certainty of the presence of the core. One of the tools developed to detect the core use this to give a parameter to measure the quality of the core detection (see below).

The difference between items 1 and 2 above is explained with the help of Figure 5.1.

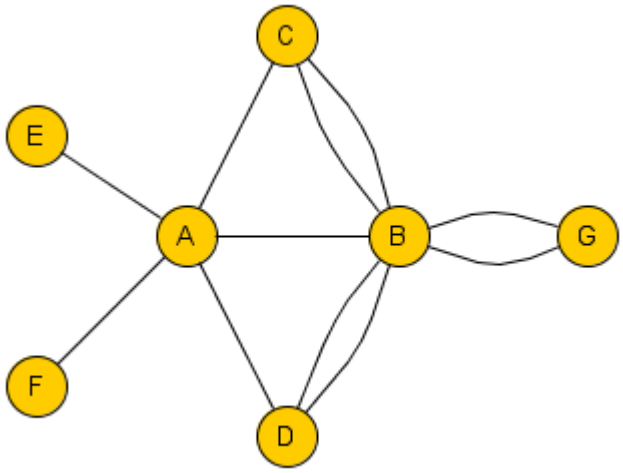


Figure 5.1 – Difference between connectivity and external degree

In Figure 5.1, the nodes represent modules and the edges are representing the connections between the modules. If any node from module **A** connects to any node in module **B**, there will be a link. The number of links between the modules reflects the total number of connections between nodes in these modules.

As can be seen in the figure, module **A** has nodes connecting to nodes in modules **B**, **C**, **D**, **E** and **F**; module **B** has nodes connecting to nodes in modules **A**, **C**, **D** and **G**.

Connections from **B** to **C**, **D** and **G** are duplicated. This means that there are 2 nodes in module **B** that connect to nodes in these other modules.

So, looking at Figure 5.1, we see that while node **B** is the one with the highest external degree (7 connections), node **A** is the one that connects to more modules (5 modules while **B** connects to only 4 modules). That's why connectivity and external degree are not the same thing.

The difference between criteria 3 and 4 is because the centrality of the module (item 4) is calculated considering modules as nodes and item 3 is the average of the centrality of the nodes in that module. These numbers can be different. For centrality calculation, closeness centrality is used (see Section 1.3.9.3).

The last item (5) is, from our experience testing different networks, related to the core module. The method for decomposition based on modularity will try to create modules in a way that the nodes inside the module will be highly interconnected and the connections between modules will be reduced (as shown in Section 4.3.3). As the core is interconnecting the modules in the periphery, this rule doesn't work well with the core module and it is allowed to have a higher external degree. That's why this ratio should be higher in the core then in other modules.

Based on these criteria, a tool was developed to evaluate the decomposition of a network and find out if there is a core present or not (see Section 3.2.6). The output of this tool can be seen in Appendix B for *E. coli* and *B. subtillis* networks.

This tool will analyze each module searching for the characteristics of a core module and then will weight these characteristics to give a parameter that shows the certainty degree of that network having a core. The higher weights are given to connectivity and external degree and the smaller ones to centrality and degree ratio. One example of the results of this tool is shown in Table 5.3 and the output of the tool can be seen in Appendix B.

5.2.2 Without clustering

Is it possible to detect a core in a network without clustering it first? After clustering, one can select, from the available modules, which one is the core as shown in Section 5.2.1, but how to do this before clustering? Here, two methods to achieve this are proposed.

5.2.2.1 Using Capacity

Capacity is described in Section 1.3.10 and it is a way to measure the robustness of a network. As the core connects different parts of the network, by the removal of the nodes in the core, there should be a decrease in the capacity of a network.

To test the parameter, we remove the nodes using the same criteria used to extract the core (see Section 5.4.1) and calculate the capacity of the network after the removal of each node. The results for *E. coli* are shown in Figure 5.2 and to a random network based on the model from Barabási-Albert (Barabasi & Albert 1999) in Figure 5.3. The Barabási-Albert model was used to show the change in capacity in a network that doesn't present the core-periphery structure. For more examples, see Appendix C.

In the examples of Figures 5.2 and 5.3, the nodes were removed in decreasing order of closeness centrality (see Section 5.4.1).

In networks which have the core-periphery structure, the removal of the core nodes will decrease the capacity because many periphery nodes will be disconnected from the network without the links from the core. The removal of the core will "break" the network.

To get a numerical parameter to detect the core-periphery structure, the *accumulated capacity* parameter was defined. It is generated from the integration of the capacity change (Figures 5.2 and 5.3), resulting in the smoother curves shown in Figures 5.4 and 5.5.

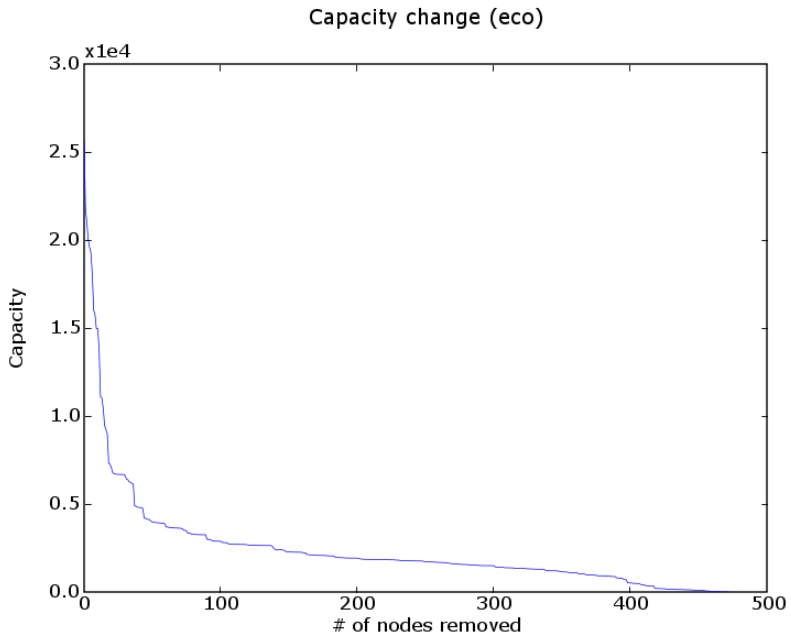


Figure 5.2 – Change in capacity for *E. coli* metabolic network by removal of most central nodes.

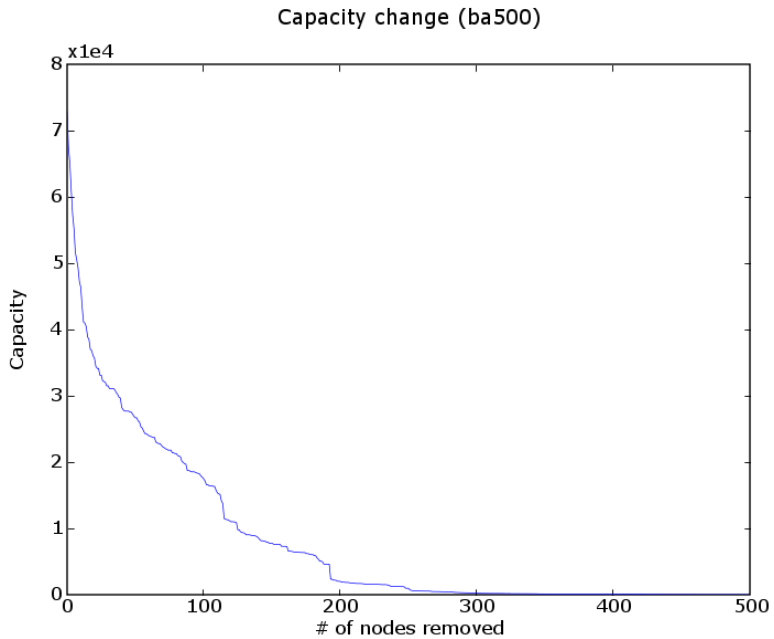


Figure 5.3 – Change in capacity for Barabási-Albert netowork by removal of most central nodes.

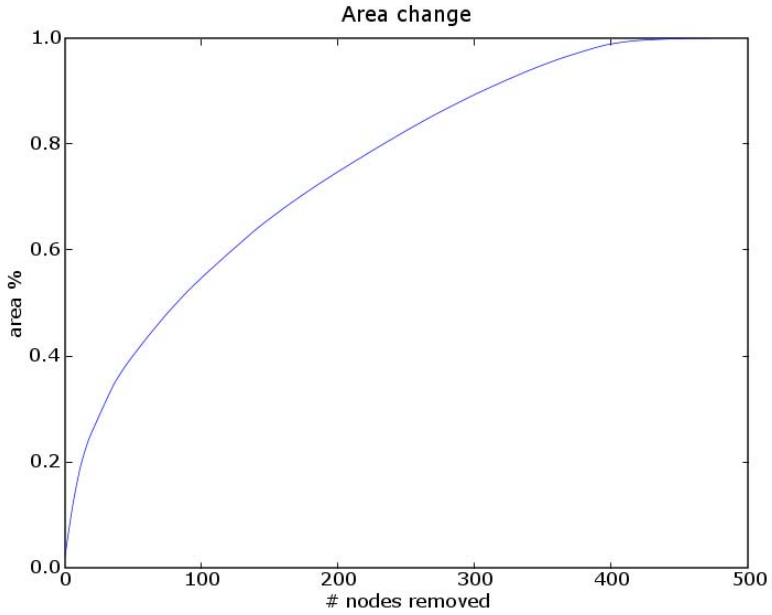


Figure 5.4 – Change in capacity for *E. coli* metabolic network by removal of most central nodes after integration.

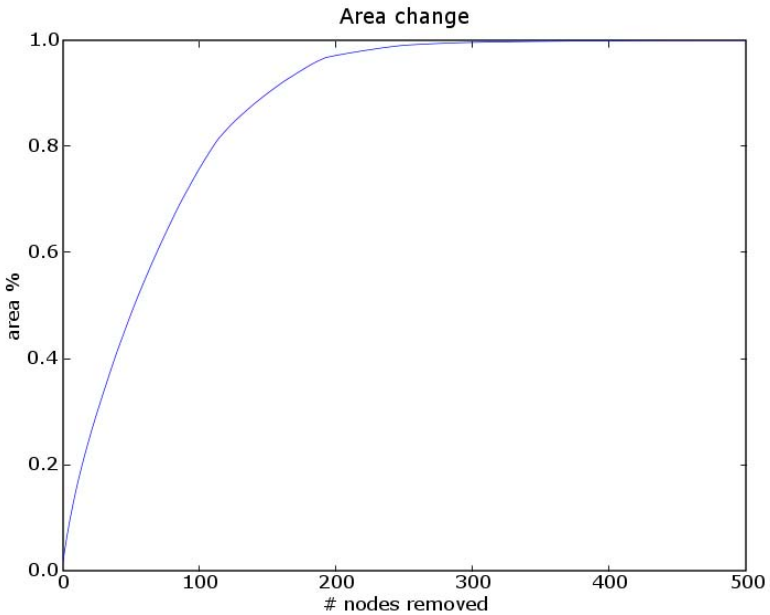


Figure 5.5 – Change in capacity for Barabási-Albert network by removal of most central nodes after integration.

From the analysis of the curves in Figures 5.4 and 5.5 (and also from the data from other organisms shown in Appendix C), we define the *core coefficient* as the percentage of nodes that need to be extracted to reach 90% of the total area in the graph of capacity (or to reach 0.9 in the accumulated capacity). It can be seen from the graph that in networks presenting a core-periphery structure, the graph reaches 90% only after more then 50% of the nodes are removed while in networks not presenting such structure, this will happens before half of the nodes where removed. This behavior can be explained by the fact that in networks with core-periphery structure, after removal of the core nodes, the network will brake into many parts (periphery modules) decreasing abruptly the capacity. In networks not presenting a core-periphery structure, the removal of the nodes will not affect capacity so quickly (because the modules are more interconnected), making its decrease slower. This way, in a network not presenting a core-periphery structure the area below the picture will be concentrated in the initial values (left part as in Figure 5.3) and its integral will increase faster as shows Figure 5.5.

Thus, the core coefficient (cc) can be defined as:

$$cc = \frac{n}{N} \tag{5.1}$$

where N is the total number of nodes in the network and n should satisfy the equation:

$$\int_{i=0}^{n} C_i = 0.9 \int_{j=0}^{N} C_j \tag{5.2}$$

or, considering that the data is discrete:

$$\sum_{i=0}^{n} C_i = 0.9 \sum_{j=0}^{N} C_j \tag{5.3}$$

where C_i is the capacity of the network after removal of i nodes.

The implementation of this parameter in software is quite simple and very fast to calculate once the values for capacity are calculated. This calculation can be made with the *verifyCoreCoefficient.py* tool included in the *libclust* package (see Section 3.2.6).

Networks with a higher core coefficient present a stronger core-periphery structure because this means that the capacity will drop faster after the removal of the central nodes. As a detection of presence/absence of core-periphery structure, a value of $cc = 50\%$ made a good detection in all our examples, as it is shown in Table 5.1. The Newman-Watss-Strogatz network (Newman *et al.* 2002) is also a computer generated network like the Barabási-Albert network (Barabasi & Albert 1999).

Table 5.1 – Core coefficient for various networks

network	*core coefficient*
E. coli	0.64693
B. subtilis	0.64608
A. pernix	0.55921
S. cerevisiae	0.51676
H. sapiens	0.51866
Barabási-Albert	0.30000
Newman-Watts-Strogatz	0.34000

5.2.2.2 Using biggest module size

An alternative to the method based on the variation in capacity to detect the presence of a core-periphery structure is to check the variation in size of the biggest connected part in the network by the removal of the most central nodes.

The advantage of this method is that it is faster to measure the size of the biggest connected part then to calculate the capacity after the removal of each node of the network.

The calculation of capacity takes in account the size of the pathways in the network (see equation 1.4). If the biggest connected part of the network changes little, the value of capacity will also not present big variations, but if the network is "broken" in the middle, both, the size of the biggest connected part and the capacity will decrease. Because of this

relation between the size of the biggest connected part in the network and the capacity, both methods present similar results as show tables 5.1 and 5.2.

For the calculation of the core coefficient, the same method from equations 5.1 to 5.3 is applied, resulting in the values from Table 5.2.

Table 5.2 – Core coefficient based on biggest connected part

network	*core coefficient*
E. coli	0.75949
B. subtilis	0.75118
A. pernix	0.69281
S. cerevisiae	0.63231
H. sapiens	0.77059
Barabási-Albert	0.37924
Newman-Watts-Strogatz	0.41118

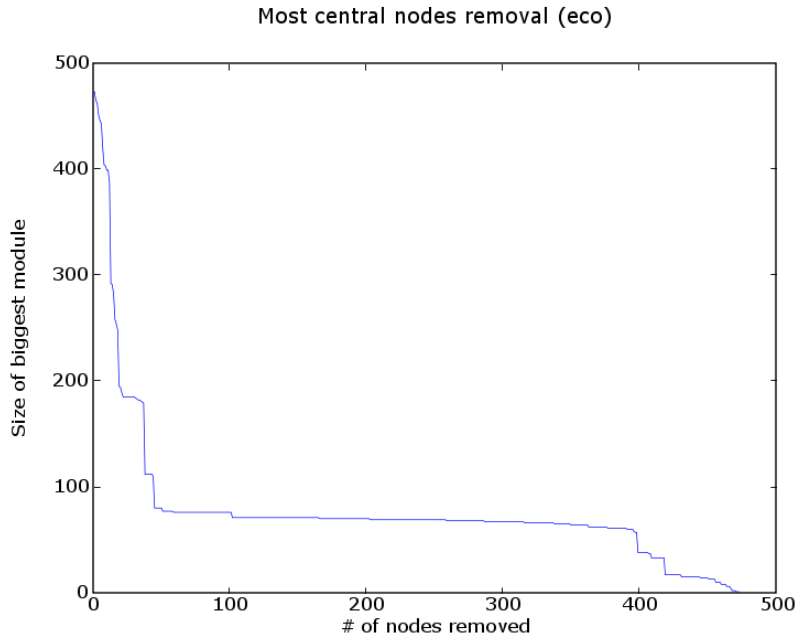


Figure 5.6 – Variation in the size of the biggest connected part of the *E. coli* metabolic network by removal of the most central nodes.

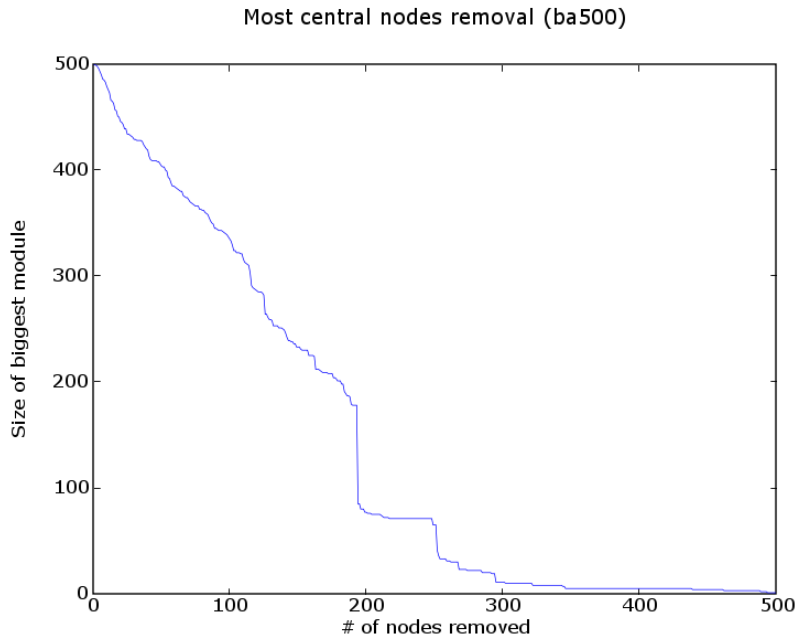


Figure 5.7 – Variation in the size of the biggest connected part of the Barabási-Albert network by removal of the most central nodes.

If the network has a core, the removal of the nodes belonging to the core will "break" the network into smaller parts (the periphery modules) that will be disconnected between each other because of the absence of the "connecting" nodes from the core.

To complement these results, Figures 5.8 and 5.9 show the change in the size of the biggest connected part for E. coli metabolic network and the Barabási-Albert network by removal of the least central nodes first. As it can be seen, the variation of the biggest connected part is linear, because the removal of the least central nodes doesn't separate the network into smaller parts. The decrease in size of the biggest connected part is just 1 which is the number of nodes removed in each step instead of having big "jumps" like when removing the most central nodes.

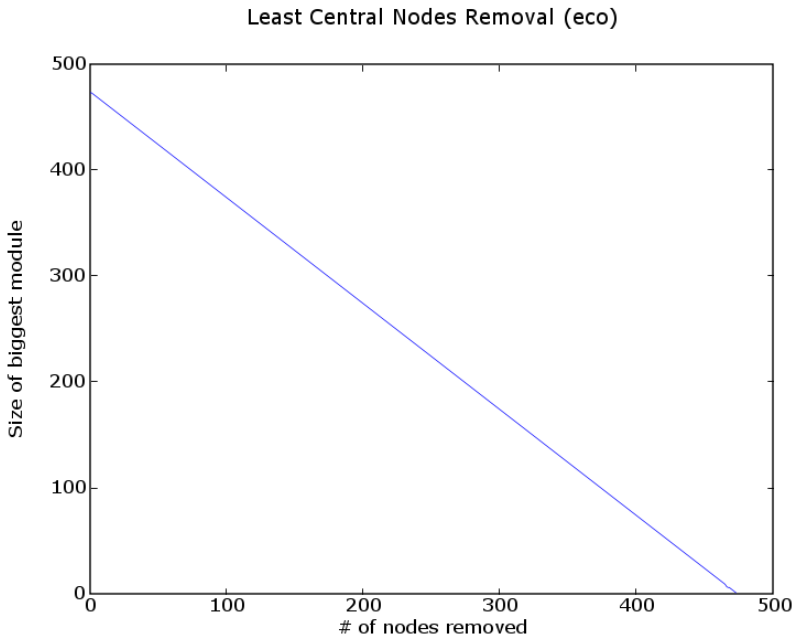


Figure 5.8 – Variation in the size of the biggest connected part of the *E. coli* metabolic network by removal of the least central nodes.

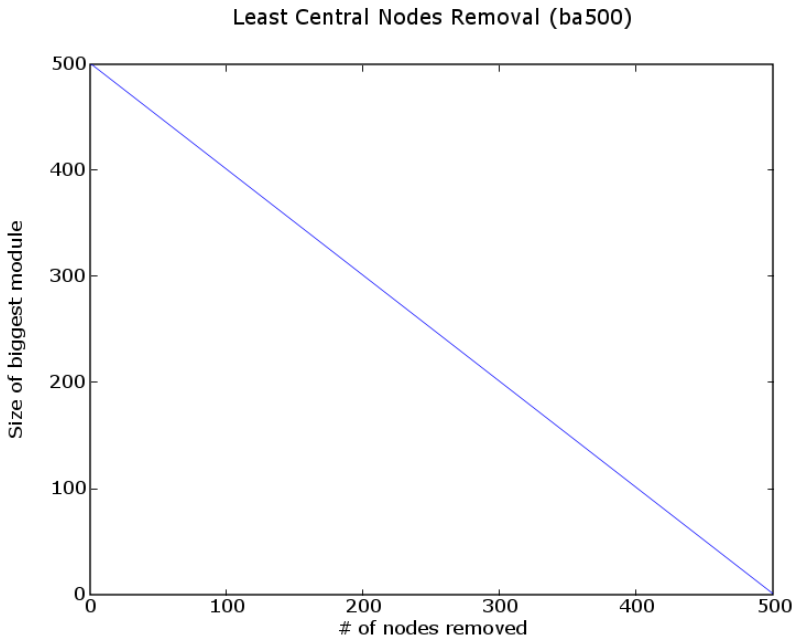


Figure 5.9 – Variation in the size of the biggest connected part of the Barabási-Albert network by removal of the least central nodes.

5.3 Using modularity to decompose a network into a core-periphery structure

To decompose a network into a core-periphery structure using the method based on modularity, we first run the decomposition algorithm without taking in account which community (module) will be the core at the end.

After decomposing the network into communities, if the core-periphery structure is present, we need to identify which one of the modules play the role as the core module.

To select the core, we use the criteria explained in Section 5.2.1. Based on these criteria, we developed a tool (*checkCore.py* – see Section 3.2.6) to find the module that represents the core using the data generated by the decomposition of the network with the modularity method. This tools evaluates the items described and will try to detect the core based on this. Table 5.3 shows some results from this tool. In Appendix B there are examples of output of the tool for *E. coli* and *B. subtilis* networks.

Table 5.3 – Core detection for decomposition based on modularity

organism	node centrality	module centrality	ext/int degree	external degree	connectivity	core guess	certainty
A. pernix	9	9/6	6	9	9	9	94.59%
B. subtilis	14	14	14	14	14	14	100.00%
E. coli	14	14	16	14/16	14	14	86.67%
S. cerevisiae	7	7	7	7	7	7	100.00%
H. sapiens	5	8	5	5	5/8	5	86.67%

After showing the values, the tool will try to guess, giving weights to the results, which module is the core.

As it can be seen from Table 5.3, the results are not unique in some networks. For *E.coli*, for example, modules 14 and 16 may be the core depending on which parameter we use for the selection of the core. If we look at the nodes in these modules, module 16 has *pyruvate* as one of its nodes and module 14 has all metabolites from the TCA cycle. So,

which one to chose? For *A. pernix*, a similar problem occurs and modules 6 and 9 are candidates to be the core.

The reason for this conflict in the criteria and separation of components of the central metabolism is that the modularity-based algorithm is not well suited for detecting core-periphery structures. This is because it tries to create the modules in a way that there will have as much as possible connections inside the module and as few as possible connections between modules (see Section 1.3.12). The core is a highly connected module. It has many connections to other modules as it interconnects the periphery modules. Because of this, the algorithm tends to split the core in many modules to better achieve this.

Figure 4.3 shows the results of the decomposition for *E. coli* metabolic network. From the picture, it is not clear which module (14 or 16) plays the role as the core. Both are interconnecting different parts of the network and are central.

For the periphery modules, the results of the decomposition from a biologic point of view are good as shown Table 4.5. Even for the core module the results are acceptable for some organisms as shows Table 5.3. From a structural point of view the problem is that there are too many connections between periphery modules as shows Figure 4.3.

5.4 Alternative method for core-periphery decomposition

As an alternative to the modularity-based decomposition algorithm, we tried another method in which we collect most central metabolites in one module, which should be the core at the end of the decomposition. This tries to fix the problem in the core detection from the original algorithm.

For this method, we choose the nodes that will make part of the core at the beginning of the algorithm and create a module. This module will take part of the remaining clustering, so other nodes can take part on this module to complete it. Because of this, the initial size of this module should not be too big.

A first attempt to this approach was unsuccessful because we were using distance to cluster the nodes after the initial module creation. The problem was that the initial module (which has many central metabolites) was highly connected and because of this its distance to other nodes was usually very short. This made the core module to grow very fast by the addition of other nodes making it, sometimes, to have more then 50% of all nodes in the network.

As a second attempt, we mixed this first approach with the traditional modularity-based algorithm. After the creation of the initial module, the remaining network is clustered using the multi-criteria method using modularity as the first criterion, which lead to good results.

5.4.1 Criterion to select the initial module

In the first step of the algorithm, it will be created an initial module which will be the core. This module should interconnect other modules. It should be central and with most of the other nodes connecting to it. To achieve this, we tested three different type of centralities (Brandes 2000) in known networks to see which one would be more successful on the identification of the core module. After the extraction of the initial core module, the algorithm runs as the normal multi-criteria algorithm, until all the nodes are clustered and the module decomposition with the highest modularity is chosen.

Degree centrality, closeness centrality and betweenness centrality (see Section 1.3.9 for definitions) were tested with the networks from Table 5.3. For *E. coli* (Figure 5.10), the core extraction using different centralities is shown in Figure 5.11 (degree centrality), Figure 5.12 (betweenness centrality) and Figure 5.14 (closeness centrality). The edge labels represent the number of edges connecting two modules (this makes the figure clearer in opposite to drawing all the edges). Node size represents the amount of nodes inside the module (bigger nodes means a higher number of nodes in the module).

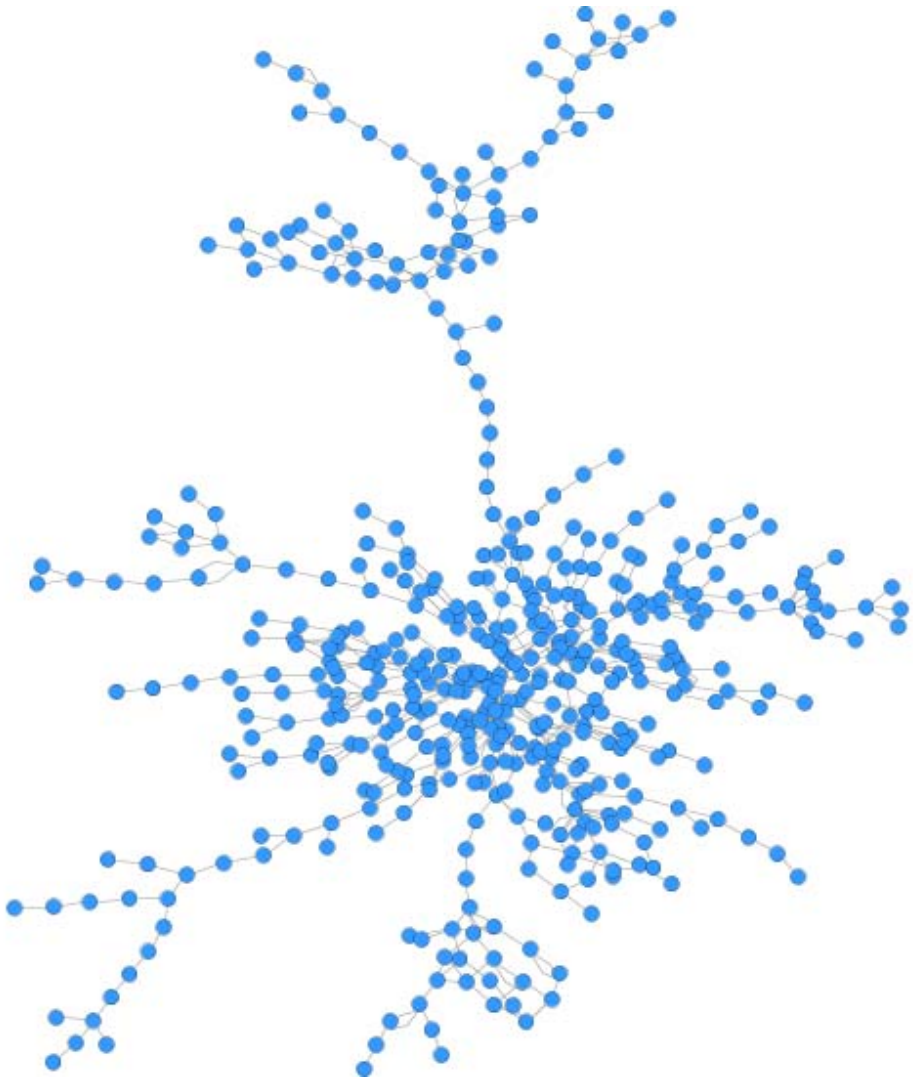


Figure 5.10 – *E.coli* metabolic network

5.4.1.1 Degree centrality

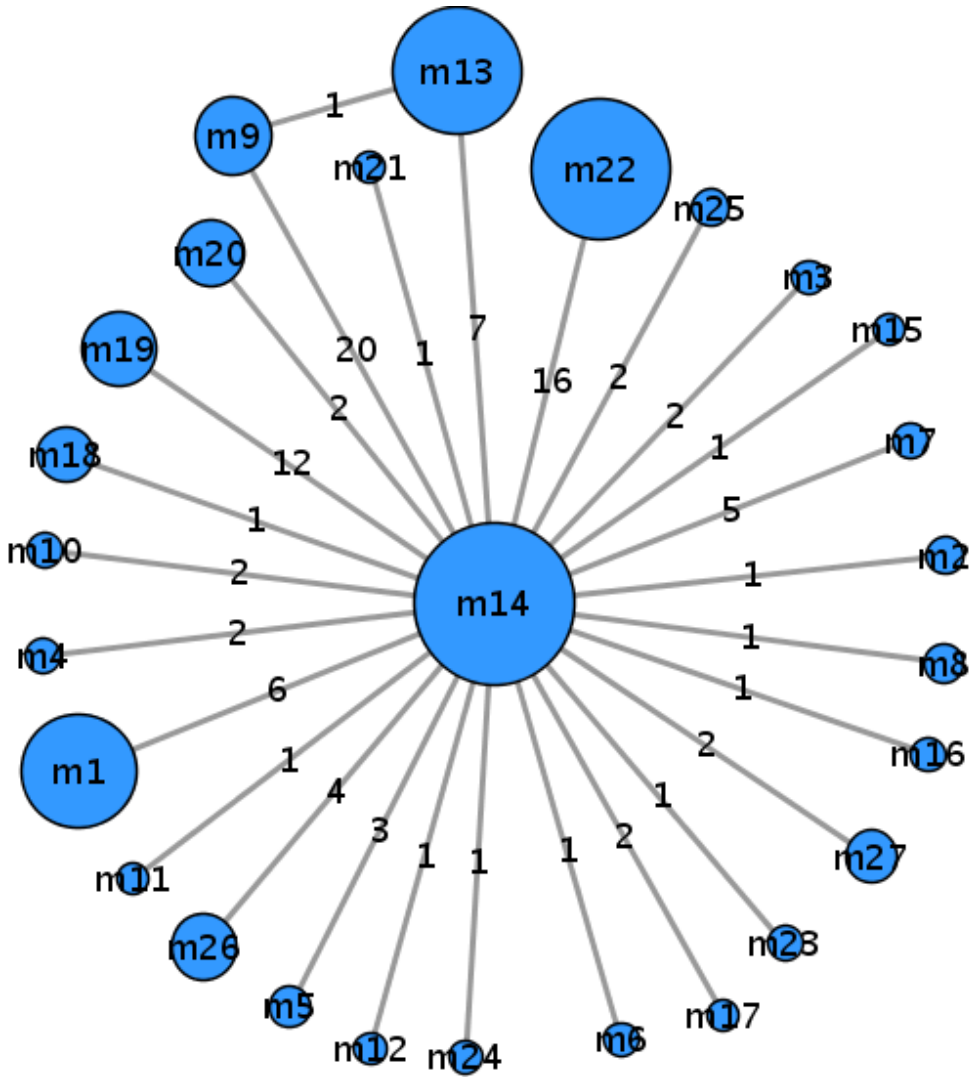


Figure 5.11 – Core extraction using degree centrality

Using degree centrality we got a picture that looks very nice, considering that all the modules connect directly to the core. But further analysis shows that it is not optimal because there are modules that are too big compared to other modules. Also, the number of connections between modules is, sometimes, too high. For example, there are 20 edges between modules 9 and 14. One way to eliminate these edges would be to join modules 9 and 14 but then the core would be too big.

For the analysis of the decomposition from a biological point of view, Table 5.4 shows the pathways that most of the metabolites in each module belong. The highlighted module is the core module. For a complete list, see Appendix D.

A problem with the use of degree centrality to select the nodes to the initial core is that there may be some highly connected nodes that will only interconnect nodes in the same module or some neighbor modules.

One example of such node is *adenine* (ADE – KEGG compound number C00147). If the nodes are ordered by their degree centrality, *adenine* is the one with the 12th highest centrality (see Table 5.9) and will be selected to be in the initial core. Looking at its position in the network (which can be measured by closeness centrality), this metabolite is not too central (position 411th in closeness centrality from a total of 473 nodes). Checking in the KEGG database, we see that this node is involved in *purine* metabolism, as well as all its neighbors. By definition, the core should have nodes that interconnect modules; if a node has all its neighbors in the same module, there is no reason for this node to be in the core. It makes more sense to keep this node together with its neighbors in the same module.

Nodes, like ADE, when "moved" to the core, will have the side effect of increasing the number of edges between this module and the core as shown in Figure 5.11, where all modules connect to the core but, sometimes, the number of connections from one module to the core may be too high.

Another side effect of this "move" from ADE to the core is that the algorithm moved other nodes from the module to the core and broke the *purine* metabolism in different modules. In the modularity-based decomposition, in the betweenness centrality and in the closeness centrality based core extraction methods, the neighbors of ADE were all in the same module (the module that has all nodes in the *purine* metabolism). Using degree centrality for core extraction this module was broken, having part of the nodes in the core and other part in the module (modules 14 and 19 respectively in Figure 5.11). This is also shown in Table 5.4, where both modules 14 and 19 have nodes from *purine* metabolism, in contrast to Table 4.4 and Table 5.6.

Table 5.4 – Pathways for module decomposition using degree centrality

Module ID	Pathways
13	Pyrimidine metabolism Arginine and proline metabolism Urea cycle and metabolism of amino groups
12	Aminosugars metabolism
11	Glycine, serine and threonine metabolism
10	Biosynthesis of steroids
17	Galactose metabolism
16	Phosphatidylinositol signaling system
15	Pentose phosphate pathway
14	Galactose metabolism Glutathione metabolism Purine metabolism Glycine, serine and threonine metabolism
19	Purine metabolism
18	Folate biosynthesis One carbon pool by folate
9	Citrate cycle (TCA cycle) Propanoate metabolism Reductive carboxylate cycle (CO2 fixation)
8	Riboflavin metabolism
7	Nucleotide sugars metabolism Starch and sucrose metabolism
6	Pentose and glucuronate interconversions
5	Fructose and mannose metabolism
4	Arginine and proline metabolism
3	beta-Alanine metabolism Alanine and aspartate metabolism
2	Nucleotide sugars metabolism
1	Lysine biosynthesis Peptidoglycan biosynthesis Methionine metabolism Glycine, serine and threonine metabolism
26	Pentose and glucuronate interconversions Pentose phosphate pathway
27	Purine metabolism Histidine metabolism
24	Benzoate degradation via CoA ligation Butanoate metabolism
25	Fatty acid biosynthesis
22	Phenylalanine, tyrosine and tryptophan biosynthesis Glycerophospholipid metabolism Glycerolipid metabolism Glycine, serine and threonine metabolism
23	Fructose and mannose metabolism
20	Pantothenate and CoA biosynthesis Valine, leucine and isoleucine biosynthesis

21	D-Alanine metabolism

5.4.1.2 Betweenness centrality

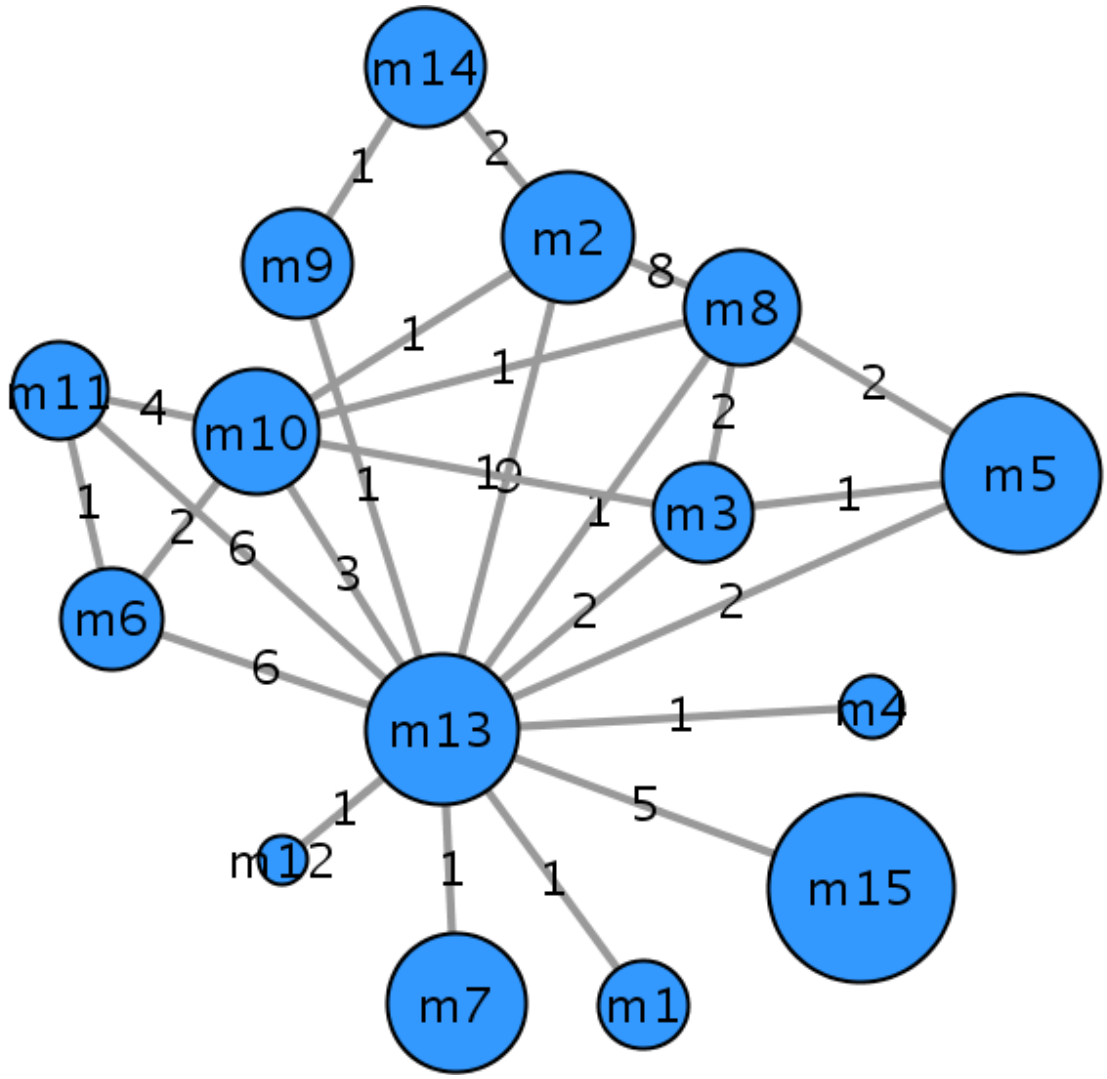


Figure 5.12 – Core extraction using betweenness centrality

For betweenness centrality, it looks like the network has no core-periphery structure at all. The modules distribution looks like in the normal method (without core extraction).

For the analysis of the decomposition from a biological point of view, Table 5.5 shows the pathways to wich most of the metabolites in each module belong. The highlighted module is the core module. For a complete list, see Appendix D.

Table 5.5 – Pathways for module decomposition using betweenness centrality

Module ID	Pathways
13	Purine metabolism Pentose phosphate pathway Carbon fixation
12	Fructose and mannose metabolism
11	Fructose and mannose metabolism Pentose and glucuronate interconversions Pyruvate metabolism
10	Glycerophospholipid metabolism Glycerolipid metabolism Glycine, serine and threonine metabolism
15	Purine metabolism Folate biosynthesis Riboflavin metabolism One carbon pool by folate
9	Pyrimidine metabolism
8	Glutathione metabolism Glycine, serine and threonine metabolism Propanoate metabolism Fatty acid biosynthesis
7	Galactose metabolism Nucleotide sugars metabolism Starch and sucrose metabolism
6	Pentose and glucuronate interconversions Pentose phosphate pathway
5	Lysine biosynthesis Peptidoglycan biosynthesis Methionine metabolism
4	Pentose and glucuronate interconversions Starch and sucrose metabolism
3	Pantothenate and CoA biosynthesis Valine, leucine and isoleucine biosynthesis
2	Citrate cycle (TCA cycle) Glyoxylate and dicarboxylate metabolism Reductive carboxylate cycle (CO2 fixation)
1	Phenylalanine, tyrosine and tryptophan biosynthesis Phenylalanine metabolism
14	Arginine and proline metabolism Urea cycle and metabolism of amino groups

The extraction of the core based on betweenness centrality will collect in the initial core, the nodes that have the highest number of shortest pathways going through them. This seems OK at a first look and may work for non-biological networks. With metabolic

networks, the most active pathways are not necessarily the shortest ones. Sometimes some longer pathways will attend more reactions and, because of this, will be more active.

One example of this is 5-phosphoribosylamine (KEGG compound number C03090) which interconnects *purine* and *glutamate* metabolism. If the nodes are ordered by their betweenness centrality, *5-phosphoribosylamine* will be the one in position 4. Only 3 metabolites have a higher betweenness centrality than *5-phosphoribosylamine*. Looking at its position in the network (which can be measured by closeness centrality), this metabolite is not too central (position 165th in closeness centrality).

All neighbors from *5-phosphoribosylamine* are classified in the same module *5-phosphoribosylamine* belongs. By definition, the core should have nodes that interconnect modules; if a node has all its neighbors in one module, there is no reason for this node to be in the core. It makes more sense to keep this node together with its neighbors in the same module.

This effect is seen by the fact that betweenness centrality didn't do a good job in identifying the central metabolites as shown in Table 5.5. Also from Table 5.5 one can see that, like in the use of degree centrality for initial core extraction, for betweenness centrality the *purine* metabolism was also "broken" – this time between modules 13 (the core) and module 15. The move of *5-phosphoribosylamine* to the core by the betweenness centrality extraction has a similar side effect like the move of ADE for degree centrality.

Another example that shows why the shortest path in a network is not always the most used path is in the conversion of *glutamate* into *arginine*. Figure 5.13 shows a picture of the pathways involved.

From Figure 5.13 we see that the shortest path from *glutamate* to *arginine* is the **B** pathway. But this pathway is not used to convert *glutamate* into *arginine*. The initial part of this path is used for *proline* synthesis. When *L-glutamate 5-semialdehyde* is produced

it is first converted to *L-1-pyrroline-5-carboxylate* without the help from any enzyme and it is later converted into *proline*, following the **C** path in the picture.

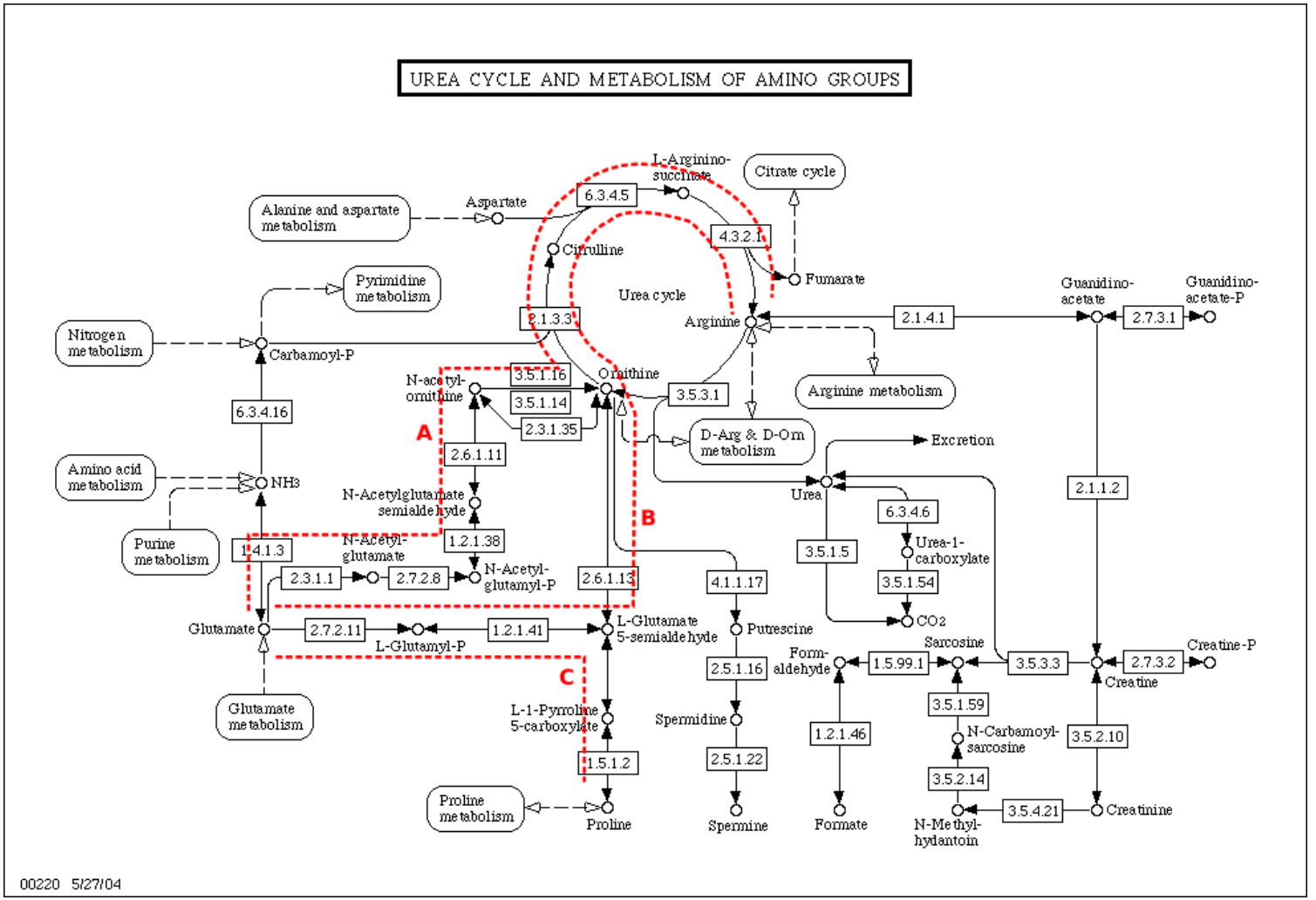


Figure 5.13 – Conversion from *glutamate* to *arginine*.

The **A** path is longer then the **B** one, but it is the one actually used to convert *glutamate* into *arginine*.

5.4.1.3 Closeness centrality

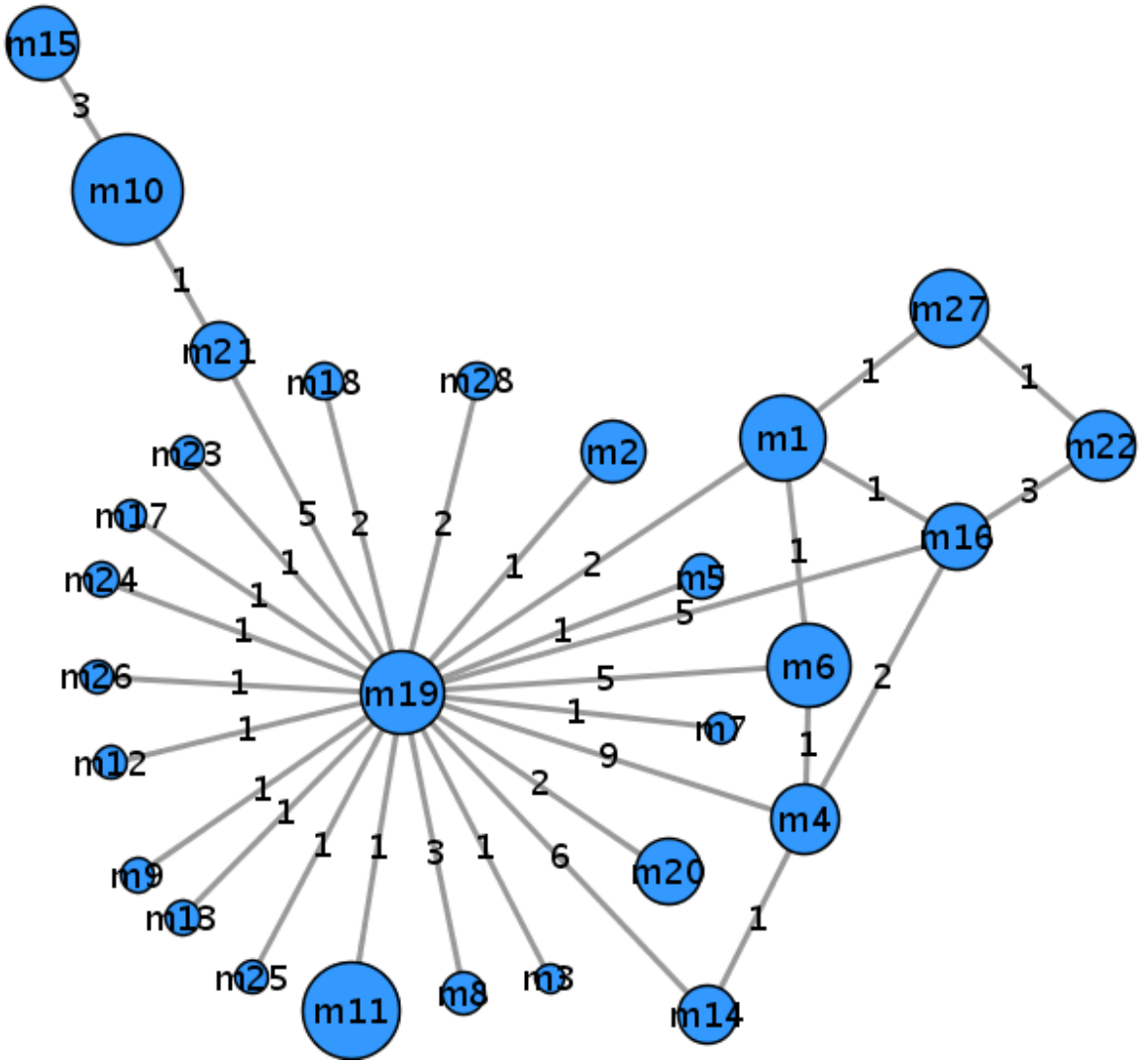


Figure 5.14 – Core extraction using closeness centrality

The decomposition using closeness centrality for core extraction shows a structural result between the other two (using betweenness and degree centrality). The core-periphery structure seems to be present but some modules are not connected directly to the core, like modules 10, 15, 22 and 27. Figure 5.15 shows these nodes highlighted in the network together with the modules that link them to the core. The yellow square nodes belong to

modules 10 and 15 (not connected to the core); nodes from module 21 are light green triangles; nodes with a green round rectangle shape are nodes in modules 1, 4, 6, 14 and 16; and red diamonds nodes are part of modules 22 and 27 (not connected to the core).

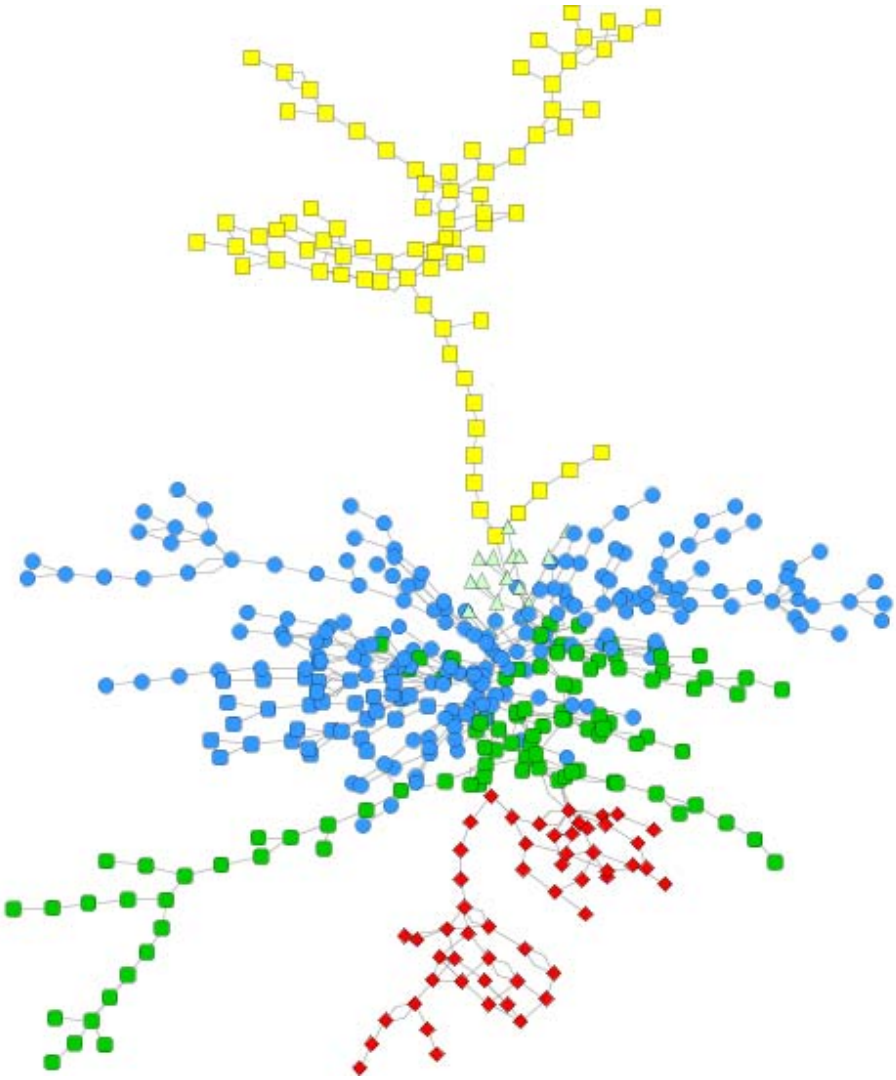


Figure 5.15 – *E. coli* network with modules 1, 4, 6, 10, 14, 15, 16, 21, 22 and 27 highlighted

Figure 5.16 shows the detail of the connection between modules 1 (yellow square), 16 (light green triangles), 22 (blue circles) and 27 (red diamonds). From the picture one can see that module 27 is a branch with highly connected community of nodes in the final part. Module 22 also shows a highly connected community with only three connections to module 16. From the structural organization of the network is acceptable that these

modules are not directly connected to the core to keep the organization in communities. The structure from modules 10, 15 and 21 shows a similar branch like module 27. This can be seen in Figures 5.10 and 5.14.

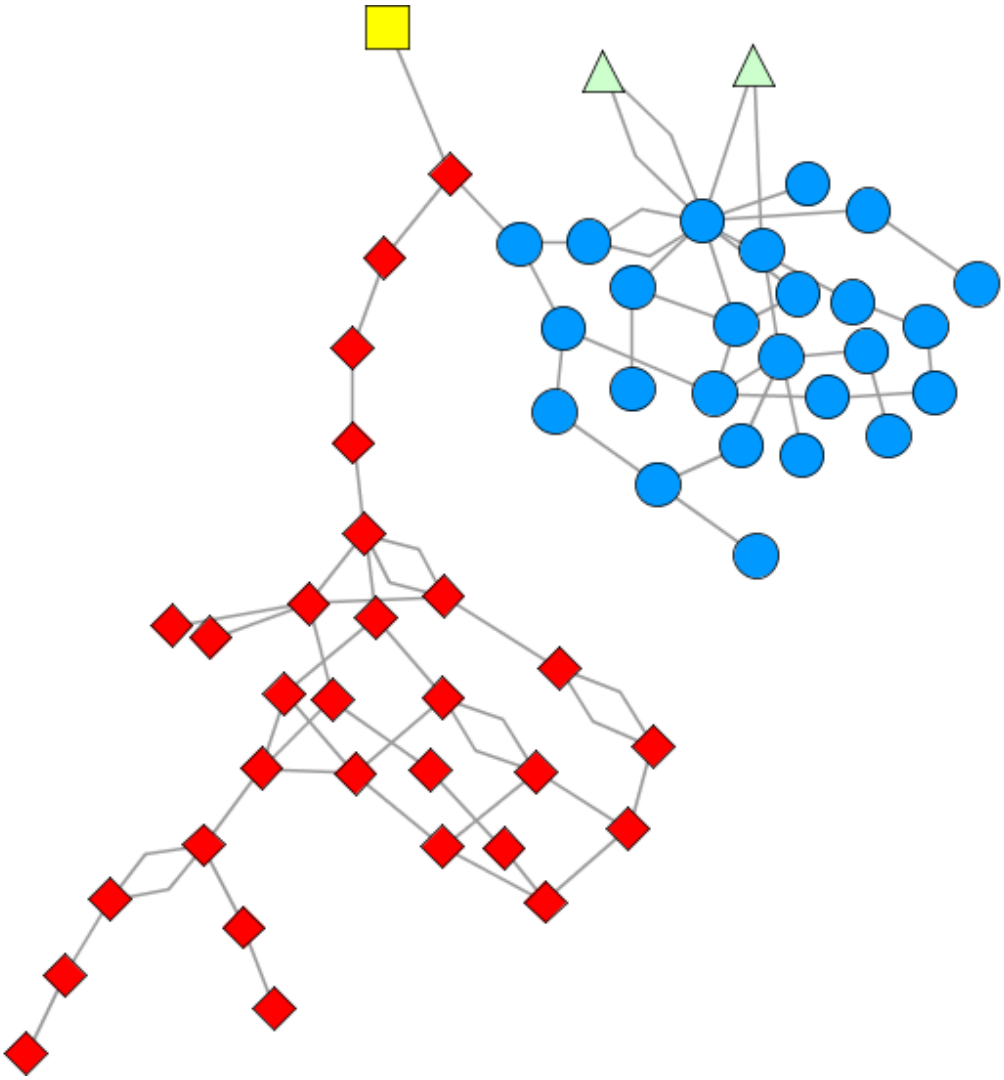


Figure 5.16 – Detailed connection of modules 22 and 27 to modules 1 and 16

For the analysis of the decomposition from a biological point of view, Table 5.6 shows the pathways that most of the metabolites in each module belong. The highlighted module is the core module. For a complete list, see Appendix D. From this table is possible to see that the core was well identified (module 19) having most of its nodes in *pyruvate* metabolism and *glycolysis*. For other modules the results are also good.

Table 5.6 – Pathways for module decomposition using closeness centrality

Module ID	Pathways
13	Valine, leucine and isoleucine degradation Valine, leucine and isoleucine biosynthesis
12	Aminosugars metabolism
11	Galactose metabolism Nucleotide sugars metabolism
10	Purine metabolism
17	Pentose phosphate pathway
16	Histidine metabolism Citrate cycle (TCA cycle) Glutamate metabolism Butanoate metabolism
15	Folate biosynthesis Riboflavin metabolism One carbon pool by folate
14	Glycerophospholipid metabolism Glycerolipid metabolism
19	Pyruvate metabolism Glycolysis / Gluconeogenesis
18	Propanoate metabolism Valine, leucine and isoleucine degradation
9	Pentose and glucuronate interconversions Glyoxylate and dicarboxylate metabolism
8	Fructose and mannose metabolism
7	D-Alanine metabolism
6	Glycine, serine and threonine metabolism Methionine metabolism Glutathione metabolism
10	Pentose and glucuronate interconversions
4	Glyoxylate and dicarboxylate metabolism Glycine, serine and threonine metabolism
3	Glycine, serine and threonine metabolism
1	Lysine biosynthesis Peptidoglycan biosynthesis Aminosugars metabolism beta-Alanine metabolism Alanine and aspartate metabolism
26	Benzoate degradation via CoA ligation
27	Pyrimidine metabolism
25	Fructose and mannose metabolism
22	Urea cycle and metabolism of amino groups Arginine and proline metabolism
23	Arginine and proline metabolism

20	Pantothenate and CoA biosynthesis Valine, leucine and isoleucine biosynthesis Valine, leucine and isoleucine degradation
21	Pentose and glucuronate interconversions Pentose phosphate pathway Carbon fixation
28	Fatty acid biosynthesis

5.4.1.4 Summary

In the last 3 sections a comparison of the decomposition method by initial core extraction using three different types of centralities for selecting the initial core was shown.

From the structural point of view, closeness centrality looked more reasonable, generating modules with similar sizes and most of the modules connected to the core module. There are exceptions (modules 10, 15, 22, 27), but as explained before, it is acceptable.

From the biological point of view, we see that most of the nodes in the core module created by the closeness centrality method are located in the *pyruvate* metabolism and *glycolysis* (see Table 5.6) while with betweenness and degree centralities these pathways are not listed in the core. This means that these two pathways have been separated in other modules. As complementary information, Tables 5.7 to 5.9 show the twelve metabolites from the *E. coli* metabolic network with highest betweenness, closeness and degree centrality. The metabolites that belong to the central metabolism are marked.

As it can be seen from Tables 5.7 to 5.9, while not perfect, closeness centrality does a better job than the other two in finding the central metabolism.

From these results, it was decided to use closeness centrality in the cluster tool for the initial extraction of the core. Betweenness and degree centrality are also available by the use of **`--centr`** option of the tool (see Appendix B).

Table 5.7 – 12 metabolites with highest betweenness centrality for *E. coli* network.

#	betweeness centrality	central met.	KEGG number	name
1	0.44269532	X	C00022	Pyruvate
2	0.28220324	X	C00117	D-Ribose 5-phosphate
3	0.27042175		C00119	5-Phospho-alpha-D-ribose 1-diphosphate
4	0.25315772		C03090	5-Phosphoribosylamine
5	0.25016193		C03838	5'-Phosphoribosylglycinamide
6	0.24714815		C04376	5'-Phosphoribosyl-N-formylglycinamide
7	0.24411638		C04640	2-(Formamido)-N1-(5'-phosphoribosyl)acetamidine
8	0.24106661		C03373	Aminoimidazole ribotide
9	0.23799885		C04751	1-(5-Phospho-D-ribosyl)-5-amino-4-imidazolecarboxylate
10	0.23491310		C04823	RSCAI
11	0.23236712		C04677	AICAR
12	0.22852474		C00130	IMP

Table 5.8 – 12 metabolites with highest closeness centrality for *E. coli* network.

#	closeness centrality	central met.	KEGG number	name
1	0.13162298	X	C00022	Pyruvate
2	0.12777477	X	C00074	PEP
3	0.12770563	X	C04442	2-Dehydro-3-deoxy-6-phospho-D-gluconate
4	0.12437418		C11437	Phthalazin-1-one
5	0.12424322		C01286	2-Dehydro-3-deoxy-D-galactonate 6-phosphate
6	0.12404731		C04691	DAHP
7	0.12401471	X	C00111	Glycerone phosphate
8	0.12385201	X	C00149	Malate
9	0.12356021	X	C00118	D-Glyceraldehyde 3-phosphate
10	0.12310902		C00546	MGLO
11	0.12183789	X	C00024	Acetyl-CoA
12	0.12155550	X	C00279	E4P

Table 5.9 – 12 metabolites with highest degree centrality for *E. coli* network.

#	degree centrality	central met.	KEGG number	name
1	0.04237288	X	C00022	Pyruvate
2	0.02542373	X	C00024	Acetyl-CoA
3	0.02330508	X	C00111	Glycerone phosphate
4	0.01906780		C00025	Glutamate
4	0.01906780		C00124	D-Galactose
4	0.01906780	X	C00118	D-Glyceraldehyde 3-phosphate
7	0.01694915		C00031	D-Glucose
7	0.01694915	X	C00092	D-Glucose 6-phosphate
7	0.01694915	X	C00085	D-Fructose 6-phosphate
10	0.01483051		C00051	GLT
10	0.01483051		C00049	Aspartate
12	0.01271186		C00147	ADE

Chapter 6
Final discussions and future work

The results for the decomposition of different networks using the tools developed were presented in this work. Analysis of the results shows that the results are satisfactory both from a structural (mathematical) point of view as well as from a biologic point of view as the modules show metabolites from the same or related pathways. For a complete list of results with pictures, see Appendix D.

While modularity gives very good results in the classification of the periphery network it has, sometimes, problems in the central part. As a solution for this problem a core-periphery approach for the decomposition was also presented with good results in the localization of the central metabolites. For the periphery part both methods presented similar results as the core-periphery-based method also uses modularity after the initial core extraction.

In the following sections some interesting characteristics of the networks studied and limitations in the graph representation are presented. Suggestions of work that can be derived from this one are also discussed.

6.1 Fragility versus Centrality

Fragility of a node (Section 1.3.11) shows how much the network capacity would be affected by the removal of that node. Centrality of a node (Section 1.3.9) shows the importance of that node in the network, based on its position (closeness centrality), the number of shortest paths that go through it (betweenness centrality) or the connectivity of that node (degree centrality). One would expect that the most important nodes in a network should also be the most fragile ones. If these important nodes are removed, the capacity of the network should decrease.

In the metabolic networks analyzed, this is not true as shows Table 6.1 for *A. pernix* and Table 6.2 for *B. subtilis*. Let's get as examples pyruvate (PYR) and acetyl-CoA (AcCoA). For *A.pernix*, while pyruvate is very central (2^{nd} for closeness centrality, 1^{st} for betweenness and degree centrality) it is the 5^{th} most fragile metabolite. There are 4 metabolites more fragile then pyruvate. For acetyl-CoA, also a very central metabolite (6^{th} in closeness centrality) and highly connected (2^{nd} in degree centrality) is the 11^{th} metabolite most fragile. In the case of *B. subtilis*, pyruvate is the most central metabolite based on the three centralities, but it is the 8^{th} most fragile metabolite. Also acetyl-CoA has a low fragility (15^{th} in the list). This is also true for other networks as, for example, *H. sapiens* where pyruvate is the 17^{th} most fragile node and acetyl-CoA is the 8^{th} most fragile node.

Table 6.1 – Fragility vs. Centrality for *A. pernix*.

A. pernix									
Compound	Name	Fragility	Pos.	Closeness centrality	Pos.	Betweenness centrality	Pos.	Degree centrality	Pos.
C00049	ASP	0.231460	1	0.167406	4	0.385324	2	0.033113	9
C00438	CAASP	0.201663	2	0.151606	14	0.317086	5	0.019868	34
C04691	DAHP	0.142895	3	0.163952	8	0.281133	6	0.019868	42
C00337	DHORO	0.139312	4	0.136528	32	0.231347	7	0.013245	51
C00022	PYR	0.132454	5	0.174769	2	0.393944	1	0.052980	1
C00295	ORO	0.130425	6	0.123974	59	0.221457	8	0.013245	73
C00105	UMP	0.125410	7	0.104282	95	0.206976	10	0.019868	32
C01103	OMP	0.123313	8	0.113363	81	0.211391	9	0.013245	96
C00111	DHAP	0.123285	9	0.130060	43	0.168160	15	0.046358	3
C00944	dHQU	0.115102	10	0.144775	22	0.169360	14	0.013245	106
C00024	AcCoA	0.114694	11	0.164130	6	0.184520	11	0.046358	2
C02637	dHSHK	0.105807	12	0.129392	48	0.159470	16	0.019868	40
C00026	AKG	0.101825	13	0.129614	47	0.116284	22	0.026490	10
C00074	PEP	0.097093	14	0.179762	1	0.381240	3	0.026490	22
C00493	SHK	0.087359	15	0.116602	68	0.136865	19	0.019868	25

Table 6.2 – Fragility vs. Centrality for *B. subtilis*.

B. subtilis									
Compound	Name	Fragility	Pos.	Closeness centrality	Pos.	Betweenness centrality	Pos.	Degree centrality	Pos.
C00085	F6P	0.195776	1	0.115195	10	0.225755	18	0.019048	5
C00117	R5P	0.152762	2	0.105369	41	0.310035	2	0.014286	17
C00119	PRPP	0.137529	3	0.098707	82	0.297295	3	0.007143	133
C03090	5-Phosphoribosylamine	0.120567	4	0.092633	144	0.278441	5	0.004762	163
C03838	5'-Phosphoribosylglycinamide	0.117295	5	0.087227	178	0.275247	6	0.004762	190
C04376	5'-Phosphoribosyl-N-formylglycinamide	0.114476	6	0.082385	220	0.272031	7	0.004762	256
C04640	2-(Formamido)-N1-(5'-phosphoribosyl)acetamidine	0.112020	7	0.078023	250	0.268792	8	0.004762	270
C00022	PYR	0.111999	8	0.124260	1	0.432132	1	0.038095	1
C03373	Aminoimidazole ribotide	0.109875	9	0.074074	272	0.265530	9	0.004762	236
C04751	1-(5-Phospho-D-ribosyl)-5-amino-4-imidazolecarboxylate	0.108017	10	0.070482	286	0.262246	10	0.004762	244
C04823	RSCAI	0.106446	11	0.067200	298	0.258939	11	0.004762	201
C04677	AICAR	0.106168	12	0.064191	311	0.256313	12	0.007143	103
C00130	IMP	0.104951	13	0.058832	337	0.252550	13	0.014286	13
C04734	FAICAR	0.102907	14	0.061404	324	0.248881	14	0.004762	151
C00024	AcCoA	0.089711	15	0.114691	11	0.108563	31	0.026190	2

Why this happens? Isn't centrality a good measure of the importance of the nodes? Note that values for betweenness centrality fit better to the values of fragility than the other centralities because this centrality uses the shortest pathway between nodes as its main parameter for calculating the importance of the node. Fragility is based on capacity, which also uses shortest path as a parameter for calculation. But it was shown in previous chapters that closeness centrality does a better job in identifying the central metabolism, so why the most central nodes are not the most fragile ones? According to Table 6.2, for *B. subtilis*, the 7 most fragile nodes have closeness centralities varying from 10^{th} to 250^{th} position (from 421 nodes) and degree centrality varying from 5^{th} to 270^{th}. Only betweenness centrality is closer to the values of fragility for the reason already mentioned.

The answer to this question is on the robustness of the network. The most important nodes, the most central ones, should not be the most fragile ones. If one of these important nodes is missing for any reason, the network would find an alternative pathway to compensate the absence of this metabolite.

But, if the above affirmation is true, how can be that the core detection based on capacity (Section 5.2.2.1) works? The point is that, individually, these nodes are not the most fragile ones but by the removal of many of them (like in the core extraction) this will damage the network, thus decreasing its capacity. There are "back-up" pathways for the absence of some of these nodes, but if many are removed, the back-ups are also removed.

6.2 Limitations in the graph representation

The networks analyzed in this work are originated from the work of Ma and Zeng (Ma & Zeng 2003). These networks are reconstructed in a similar way as the work of Jeong *et al.* (Jeong *et al.* 2000) except for the fact that in the work from Ma and Zeng, the currency metabolites were removed. Two metabolites are linked (have an edge) if they participate

on the same reaction. For example, the reaction from Figure 6.1 can be represented as shown in Figure 6.2

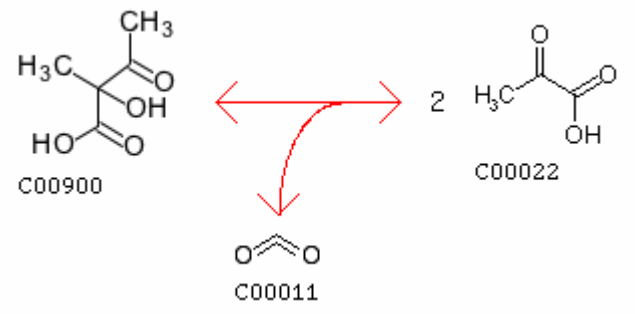


Figure 6.1 – 2-acetolactate pyruvate-lyase (carboxylating)
C00900 is the compound number for *2-acetolactate*, C00011 is the compound number for CO_2 and C00022 is the compound number for *pyruvate*.

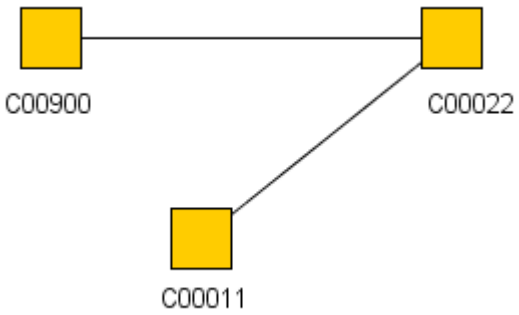


Figure 6.2 – Example of reaction represented as a graph.
C00900 is the compound number for *2-acetolactate*, C00011 is the compound number for CO_2 and C00022 is the compound number for *pyruvate*.

This procedure is done to represent all reactions in the network. In most of the cases, like the one showed above, this works pretty well. For the methods presented in this work this representation showed satisfactory results as seen in chapters 4 and 5.

In some cases though, this notation will show its limitations. For example, if we search for the shortest path between *glucose* and *pyruvate* (without considering currency metabolites), we expect to find the path shown in Figure 6.3.

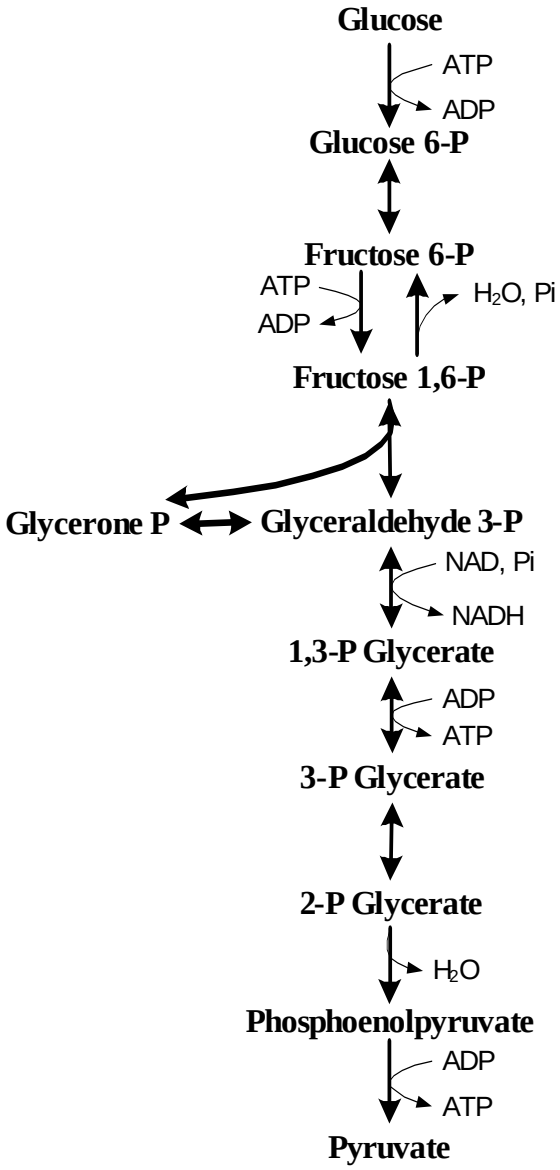


Figure 6.3 – Conversion from glucose to pyruvate

The pathway in Figure 6.3 has 9 steps from *glucose* to *pytruvate*. Calculating the shortest path for *E. coli*, instead of 9 we have 5 steps. Even without the currency metabolites, there are "shortcuts" on the pathway. For example, there is a link from *glyceraldehyde 3-P* to *pyruvate* through *2-dehydro-3-deoxy-D-galactonate 6-phosphate*. Why? Looking at

KEGG database, we see that *pyruvate* and *glyceraldehyde 3-P* are converted into *2-dehydro-3-deoxy-D-galactonate 6-phosphate* in the reaction shown in Figure 6.4.

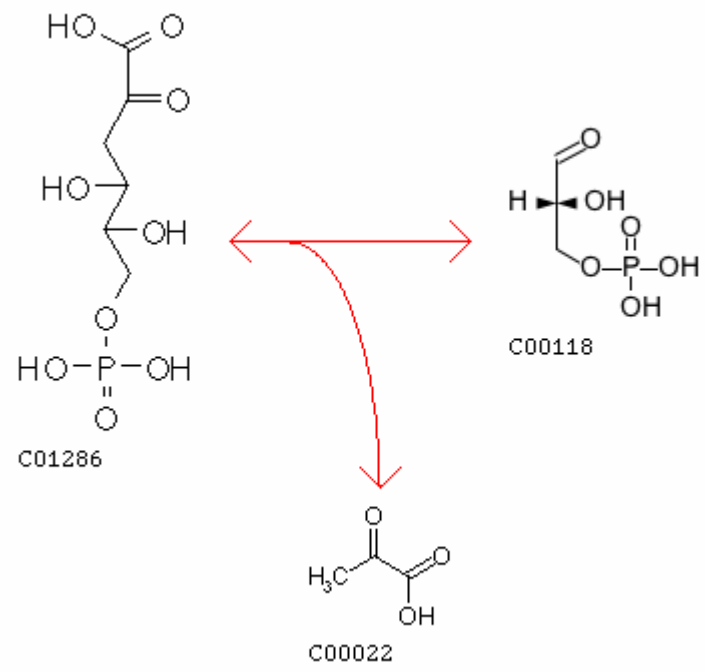


Figure 6.4 – 2-dehydro-3-deoxy-D-galactonate-6-phosphate-D-glyceraldehyde-3-phosphate-lyase
C01286 is the compound number for 2-dehydro-3-deoxy-D-galactonate 6-phosphate, C00118 is the compound number for glyceraldehyde 3-P and C00022 is the compound number for pyruvate.

As this reaction is reversible, it will add two undirected links to the network: one between C01286 and C00118 and another between C01286 and C00022. From the biologic point of view, these two links are not independent and they should have always the same direction. There should be a directed link between C01286 and C00118 and another directed link between C01286 and C00022. Another pair of directed links should exist for the opposite direction of the reaction (but not at the same time the first two links exist). The problem with the graph representation chosen is that this difference cannot be represented. Because of this, there are a link from *glyceraldehyde 3-P* to *pyruvate* through *2-dehydro-3-deoxy-D-galactonate 6-phosphate*, which is not right.

This situation is even worse if considering only the KEGG database, because many directed reactions are shown as undirected. This was corrected by the database from Ma and Zeng (Ma & Zeng 2003).

One possible solution to this problem could be the use of labels for the edges. In this case, not only the nodes have a name (representing the metabolite name), but also the edges will have the name of the reaction they represent. The reactions in the network would be represented as shown in Figure 6.5 (note that both edges receive the same label as they are representing the same reaction: R01064 in KEGG database).

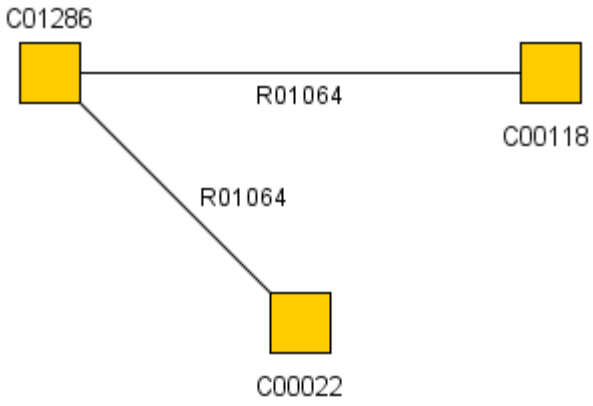


Figure 6.5 – Example of reaction represented as a graph with edge labels.

This way, the problem described in the network caused by the reaction of Figure 6.4 can be avoided if the algorithm for calculation of shortest path take in consideration that each reaction can be "followed" only once in the search for the shortest path.

6.3 Future Work

6.3.1 Fix network representation

We plan to re-create the networks studied using the proposed fix explained in Section 6.2. As the databases are updated regularly we also plan to create a tool to automate the process of reconstruction based on the ideas from Ma and Zeng (Ma & Zeng 2003), eliminating currency metabolites and fixing the direction of some reactions. To find out the currency metabolites and the reactions to be corrected we will use the database from Ma and Zeng. Another possibility for the correction of the direction of the reactions is to use the XML version (http://www.genome.jp/kegg/xml/) of the database instead of the plain text version.

6.3.2 Improvement in the cluster tool

The cluster tool works well for the decomposition of metabolic networks as shown in previous chapters. An extension of this tool is planned to work with more complex networks.

One thing to be improved is the support for networks with labeled edges like in the example of Figure 6.5. The tool should avoid crossing the same edge more then once during calculations. This should not change the behavior of the tool for networks not using edge labels.

Another improvement is to add support for integrated networks. This one is a big step for the tool as there are no good methods for this type of decomposition until now. We are developing some ideas, but these are still in very early development and should be tested with an integrated network. There is still a long way to go into this direction.

Appendix A

Links

Systems Biology group at GBF	http://www.gbf.de/SystemsBiology
Cytoscape	http://www.cytoscape.org
Cytoscape plugins	http://www.cytoscape.org/plugins2
more Cytoscape plugins	http://csresources.sourceforge.net
Network conversion tool	http://pynetconv.sourceforge.net
Python language	http://www.python.org
Java language	http://java.sun.com
Tcl/Tk language	http://www.tcl.tk

Appendix B

Output of tools

In this appendix, the output of some tools mentioned in the main text is shown.

Cluster tool version 1

Options for cluster tool version 1

```
$ cluster.py --help

usage: newCluster.py [options] network

General options:

  -h, --help            show this help message and exit.
  -s, --show-summary    shows summary of modules created (the same data saved
                        to AFILE.modules).
  -m, --use-psyco       use psyco for speed-up the execution.

Output options:

  -a AFILE, --attributes-file=AFILE
                        saves the cluster information to AFILE (in Cytoscape
                        attributes format).
```

```
  -r, --cytoscape-attribute
                        name of attribute to use as the header for Cytoscape
                        attributes file. Default: MetaData
  -t TFILE, --export-tree=TFILE
                        saves the cluster in TreeView format (only useful with
                        -C).

  --easy out            run with optimal values for easy use. Use 'out' as
                        prefix for output. Use distance in 1st stage and
                        modularity in the 2nd stage.
  --easy-mod out        run with optimal values for easy use. Use 'out' as
                        prefix for output. Always use modularity as selection
                        criterion. (much slower than --easy)

Debug options:

  -d, --debug           turns debug on.
  -g, --guess-core-size
                        used to guess the initial core size (-c) value. When
                        running in this mode there is no useful attributes or
                        tree file (so, don't use -g with -a or -t!!!).
  --no-guess-graph      do not show the guess graph at the end (but it is
                        still saved in PNG format). If this option is used,
                        the guessed value is returned as exit error code
                        (useful for batch runs).
  -T, --test-network    use test network.
  --dump-stage1         dump networks as it is after stage1

Distance calculation options:

  -D, --avg-dist        use old method for calculate distances (obsolete)

Selection criteria (note: this options change the cluster criteria, so run
                          -g option first!!!):

  --no-degree           do not use lowest degree as criteria for selection of
                        nodes to cluster.
  --no-centrality       do not use centrelity as criteria for selection of
                        nodes to cluster.
  --no-small            do not use less number of nodes as criteria for
                        selection of nodes to cluster.
  -K                    use average distance instead of centrality.

  --dist                use distance as 1st criteria (Default).
  --mod                 use modularity as 1st criteria
  --rmod                use revised modularity as 1st criteria
  --mod-rmod            use modularity until all nodes are clustered,
                        then use revised modularity
  --dist-mod            use dist until all nodes are clustered, then use
                        modularity.
  --dist-rmod           use dist until all nodes are clustered, then use
                        revised modularity.

Core options:

  -C, --no-core         do the clustering until the end, without core
                        extraction.
  -c TH1, --core-size=TH1
                        defines initial size of core (first stop of algorithm).
                        If not specified, -C is assumed.
  -G, CFILE --save-core=CFILE
```

```
                        save the core (before interface extraction) to CFILE
                        in SIF format.
  --keep-core           in stage 2, core will not cluster with peripheral
                        modules.
  --max-core-size       max. size of the core (2nd. stop)
  --min-core-size       min. size of the core (2nd. stop)
  --show-core-mod       shows modularity of free nodes.

                  Tip1: Do not use -l. -n, -N --max-mod-size, --min-mod-size
                        --max-core-size and --min-core-size together.
                        They should be used separatedly.

Final core extraction options:

  --no-ENC              do not extract isolated nodes from core to peripheral
                        modules.
  --no-FMN              do not fix modules names. Module names will remain as
                        the concatenation of the names of nodes in the module.
                        (useful for debug).
  --no-Int              do not extract core-modules interface.
  --no-CO               do not extract core-only module.
  --no-COI              do not extract core-only interface.
  -I, --no-interface    joins the interfaces with modules after extraction and
                        extract some nodes from core again.
  --no-CC               do not separate connected and unconnected core.
  --no-all-core         do all the --no-XX options except --no-FMN.
  --no-all              do all the --no-XX options.

Modules options:

  -n TH2, --number-modules=TH2
                        defines the number of modules with core (second stop).
  -N range, --nmod-range=range
                        similar to -n, but a range is informed. Use it to
                        generate a set of files with different number of
                        modules. The range should be separated with a comma
                        and no spaces are allowed. (ex. -N 10,15). Use n,-
                        for range between core estraction and n (ex. 4,-)
  -l, --no-links        stop when there are no links between peripheral
                        modules.
  -L, --to-core         stop when all modules have connection to the core.
  --max-mod-size        max. size of peripheral modules (2nd. stop)
  --min-mod-size        min. size of peripheral modules (2nd. stop)
  -H HFILE, --create-hierarchy=HFILE
                        export the modules as nodes in a SIF network when there
                        are no free nodes. The name of the generated file is
                        HFILE.m<n>.sif -- where <n> is the number of modules
                        at time of export. Assumes -C
  --H2=AFILE            pre-loads the modules using the information from the
                        AFILE attributes file.
  --show-mod            starting from when there are no free nodes, exports,
                        for each step, the attributes file and shows the
                        modularity at each step.

                  Tip1: Do not use -l. -n, -N --max-mod-size, --min-mod-size
                        --max-core-size and --min-core-size together.
                        They should be used separatedly.
                  Tip2: If no module option is given, -l will be used.

Cytoscape interaction options:
```

```
  -y START_NODE, --show-in-cytoscape=START_NODE
                        Show steps of clustering in Cytoscape starting from
                        START_NODE free nodes (requires CytoTalk plugin).
                        Negative values for -y have special meaning:
                          -1 shows extraction of some nodes from core
                          -2 shows interface extraction
                          -3 shows core-only extraction
                          -4 shows core-only interface extraction
                          -5 shows connected core extraction
  -p PORT, --cy-port=PORT
                        Use with -y if CytoTalk is listening to a port other
                        then 8082
```

Cluster tool version 2

Options for the cluster tool version 2

```
$ ncluster.py --help

usage: ncluster.py [options] network

General options:
  -h, --help            shows this help message and exit.
  -a PREFIX             sets the prefix to use in file export. Defaults to
                        base name of network.
  --core=SIZE           creates core-periphery structure instead of looking
                        for communities. Size is the initial core size. Use 0
                        to let the tool calculate it.
  --centr=C             centrality method to use with --core
                            c: closeness
                            b: betweenness
                            d: degree
  -c CRIT               sets the criteria of clustering from the pre-def. ones:
                            d: distance
                            m: modularity
                            c: centrality
                            g: degree
                            s: size
  -C CRIT               same as -c but accepts a list (one value per stage).
  -s STAGES             sets the stages from the pre-defined ones:
                            f: stop at #free nodes == 0
                            m: export modularity for each step and stop
                               at 1 module.

Extra options:
  --np                  don't show the progress bar while clustering
  -x                    export all modules composition instead of only the
                        one with best modularity
  --show-modules        while analysing the modules, show the results on screen
  --no-calc-modularity  do not calculate the modularity for every step.
  --no-save-modularity  do not save modularity at a separate file.
  --log                 save some debug info.

Advanced options:
  --config FILE         loads FILE with extra configuration.
```

Output of `checkCore.py` for *B. subtilis* data

```
Network: bsu_connected.sif
  Nodes: 421
  Edges: 498

Calculating the distance matrix and centrality...
Calcularing distance matrix for modules...

module        nodeCentr        modCentr   iDeg eDeg connect eD/iD
-------------------------------------------------------------------------
module3     0.0422577818    0.2711864407   27    1    1    0.0370370370
module6     0.0510048247    0.3636363636   42    2    2    0.0476190476
module4     0.0628309288    0.3404255319   27    1    1    0.0370370370
module5     0.0708179477    0.4571428571   21    1    1    0.0476190476
module9     0.0775062012    0.4571428571   57    6    1    0.1052631579
module13    0.0801228650    0.5000000000   41    5    3    0.1219512195
module1     0.0820853474    0.4571428571   51    2    1    0.0392156863
module12    0.0863513540    0.5161290323   24    5    2    0.2083333333
module8     0.0895263704    0.4571428571    3    1    1    0.3333333333
module17    0.0915050681    0.4571428571    8    2    1    0.2500000000
module15    0.0916766067    0.4848484848   57   12    2    0.2105263158
module7     0.0925232906    0.4571428571    6    1    1    0.1666666667
module11    0.0929848700    0.4571428571   31    8    1    0.2580645161
module16    0.0948155863    0.4571428571    3    1    1    0.3333333333
module2     0.0955985247    0.4571428571    4    1    1    0.2500000000
module10    0.1012383560    0.4571428571    2    1    1    0.5000000000
module14    0.1081370292    0.8000000000   49   40   13    0.8163265306

       Highest centrality: ['module14']
Highest module centrality: ['module14']
            highest eD/iD: ['module14']

  Highest external degree: ['module14']
     Highest connectivity: ['module14']

                    core: module14
                   score: 1.0
```

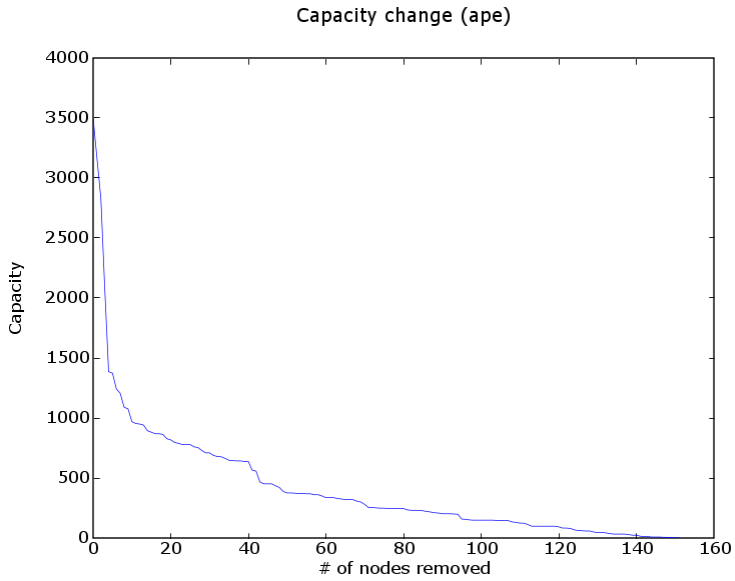
Output of `checkCore.py` for *E. coli* data

```
Network: eco-connected.sif
  Nodes: 473
  Edges: 574

Calculating the distance matrix and centrality...
Calcularing distance matrix for modules...

module        nodeCentr        modCentr   iDeg eDeg connect eD/iD
-------------------------------------------------------------------------
module4     0.0439137163    0.2542372881   38    2    1    0.0526315789
module9     0.0519033402    0.3333333333   36    3    2    0.0833333333
module5     0.0624937464    0.4285714286   25    2    2    0.0800000000
module11    0.0669538730    0.4166666667   36    2    2    0.0555555556
module3     0.0807128661    0.4166666667   38    3    2    0.0789473684
module7     0.0818973084    0.3750000000   50    1    1    0.0200000000
module2     0.0850243502    0.5357142857   32    6    5    0.1875000000
module8     0.0870296533    0.3488372093    6    1    1    0.1666666667
module15    0.0890890403    0.4054054054    5    1    1    0.2000000000
module12    0.0917713257    0.4545454545   41    5    3    0.1219512195
module6     0.0945332069    0.5172413793   51    8    5    0.1568627451
module1     0.0950493834    0.5000000000   34    7    4    0.2058823529
module13    0.0986225784    0.4054054054    3    1    1    0.3333333333
module16    0.0994842235    0.6000000000   34   16    6    0.4705882353
module10    0.1033312135    0.5769230769   46   14    6    0.3043478261
module14    0.1052121583    0.6521739130   55   16   10    0.2909090909

       Highest centrality: ['module14']
Highest module centrality: ['module14']
            highest eD/iD: ['module16']

  Highest external degree: ['module14', 'module16']
     Highest connectivity: ['module14']

Comparing ['module14', 'module16']...
             higher eD/iD: ['module16']
          higher modCentr: ['module14']

                    core: module14
                   score: 0.866666666667
for batch job:
module14 0.866666666667
```

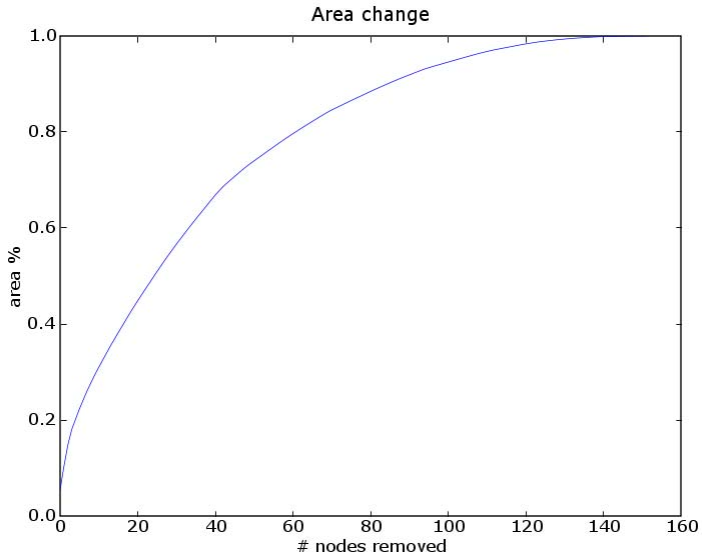
AppendixC
Core detection

In this appendix, pictures of capacity change and biggest module change for other organisms as explained in Section 5.2.2.

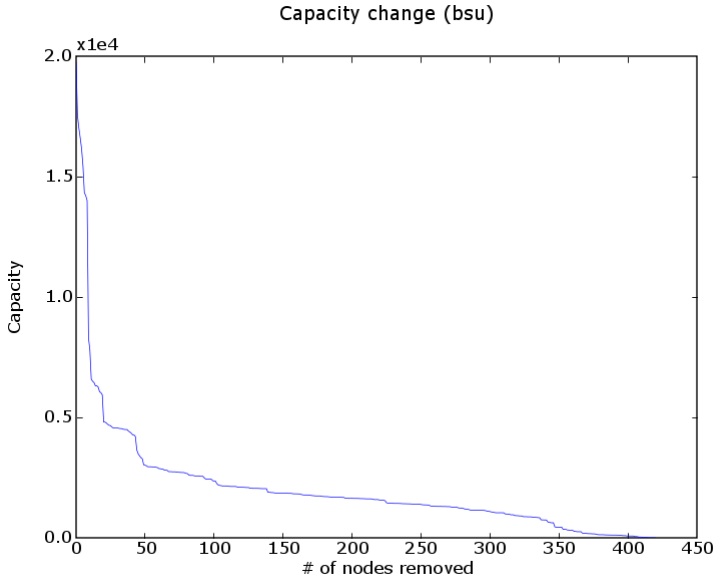
Capacity change



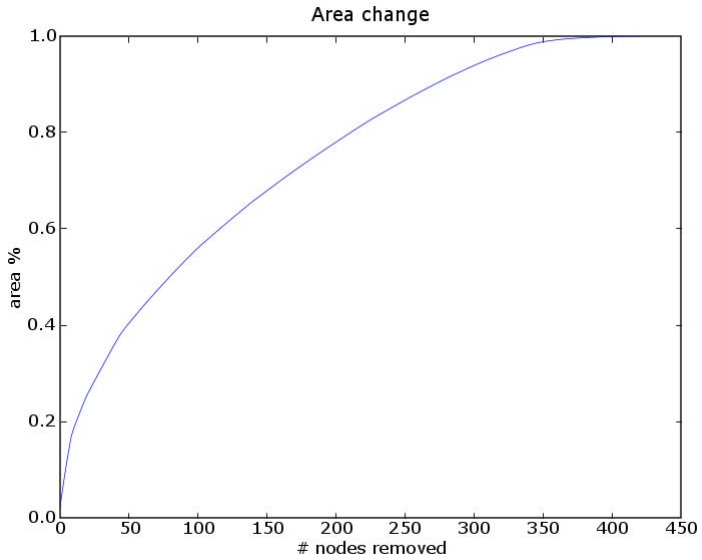
Capacity change in *A. pernix* network.



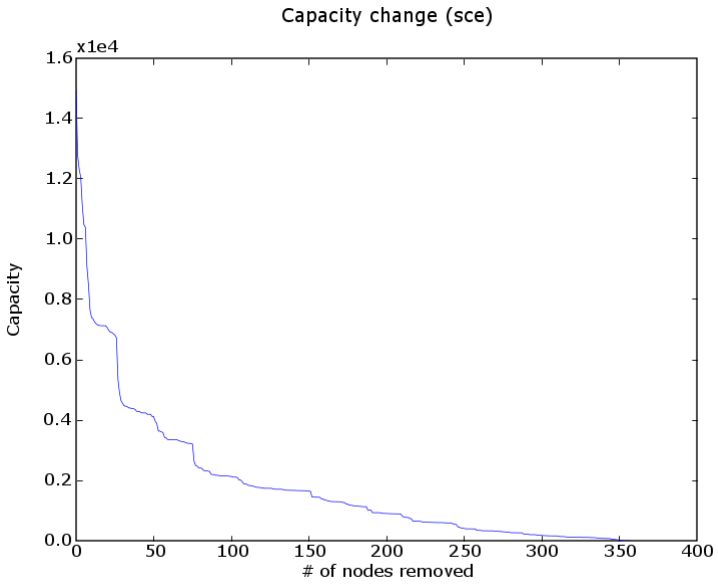
Capacity change (integrated) in *A. pernix* network.



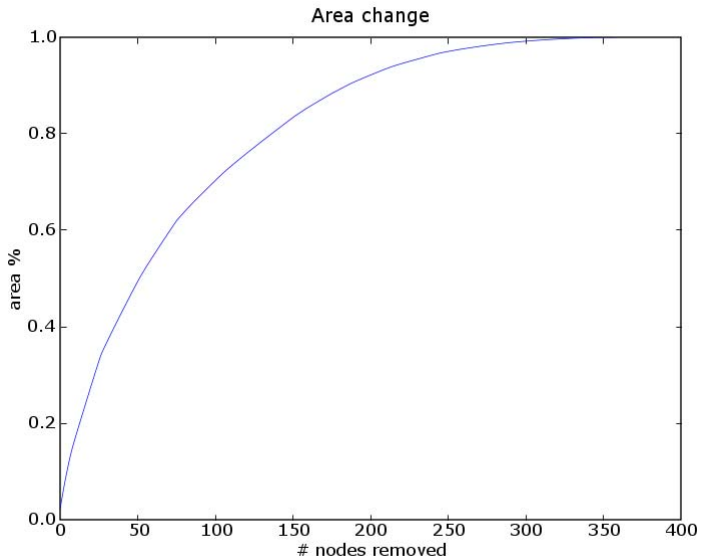
Capacity change in *B. subtilis* network.



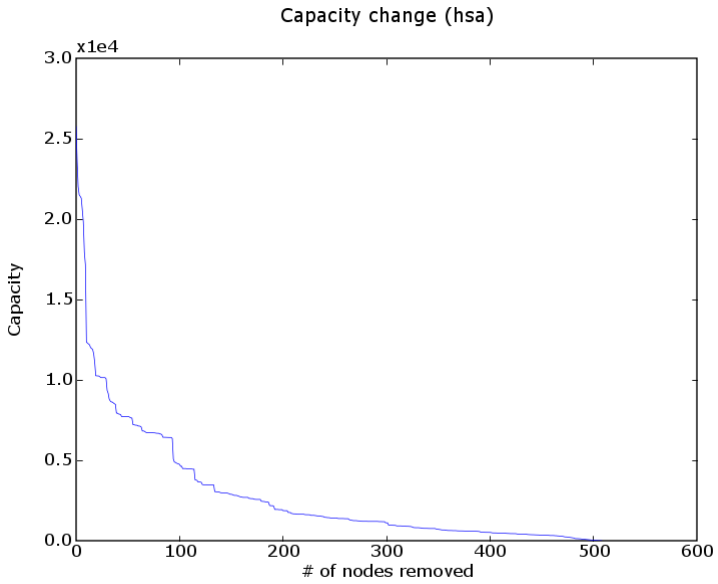
Capacity change (integrated) in *B. subtilis* network.



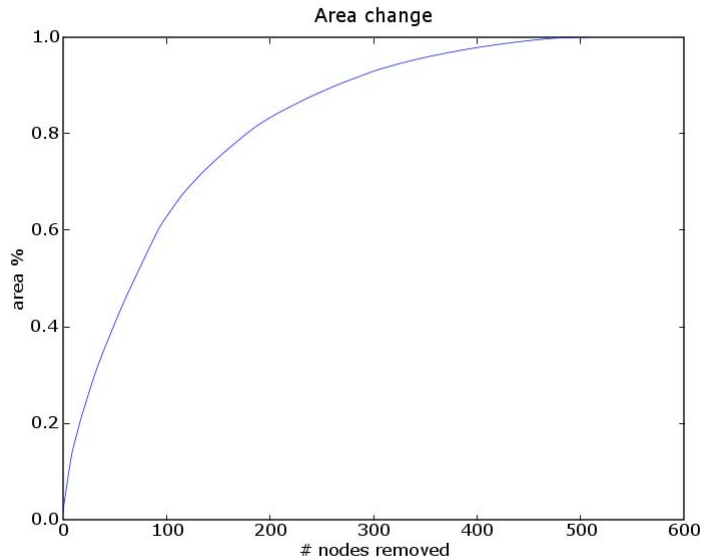
Capacity change in *S. cerevisiae* network.



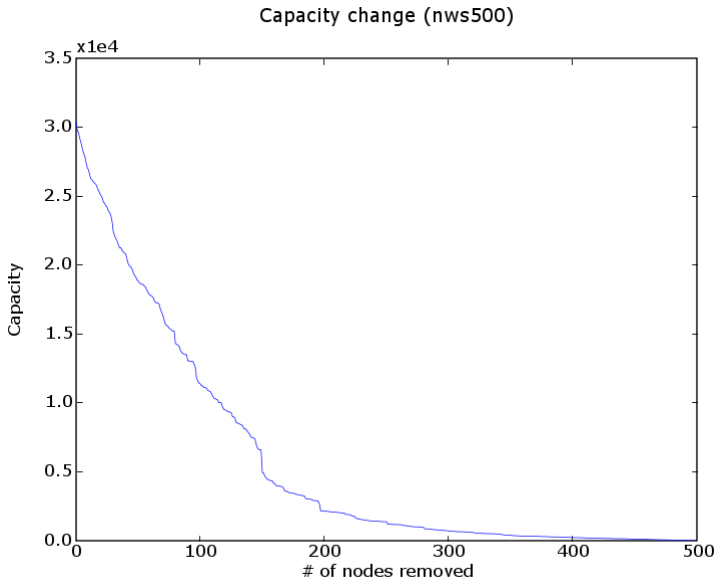
Capacity change (integrated) in *S. cerevisiae* network.



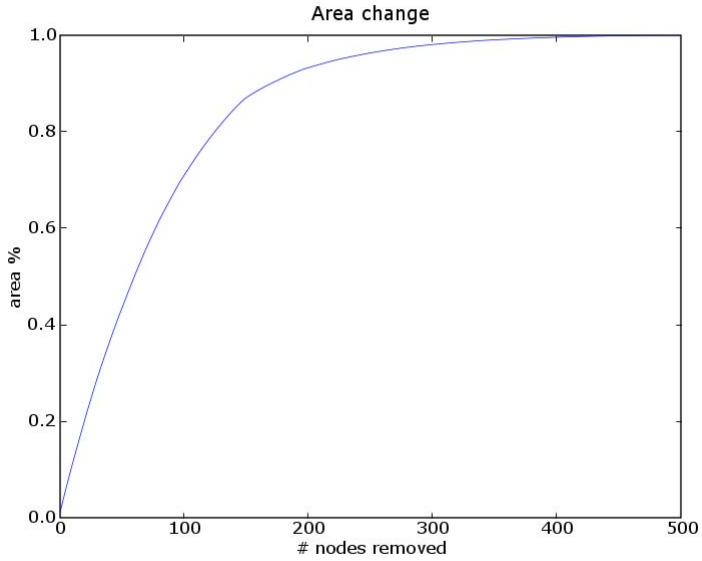
Capacity change in *H. sapiens* network.



Capacity change (integrated) in *H. sapiens* network.

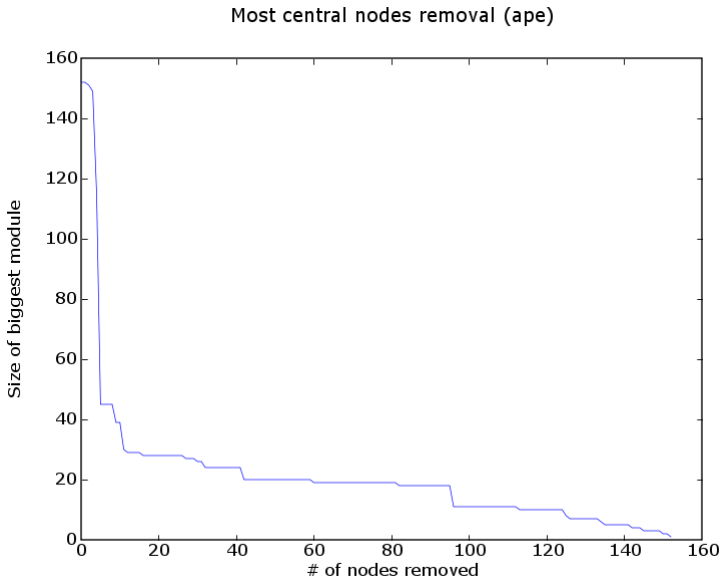


Capacity change in Newman-Watts-Strogatz network.

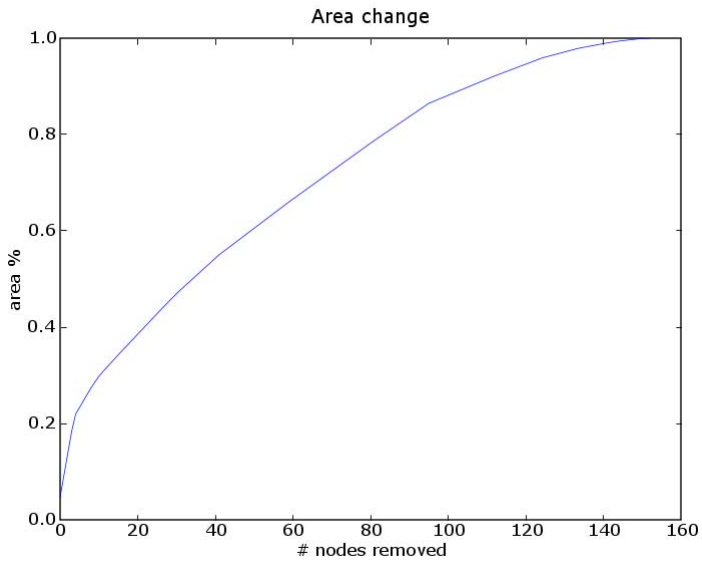


Capacity change (integrated) in Newman-Watts-Strogatz network.

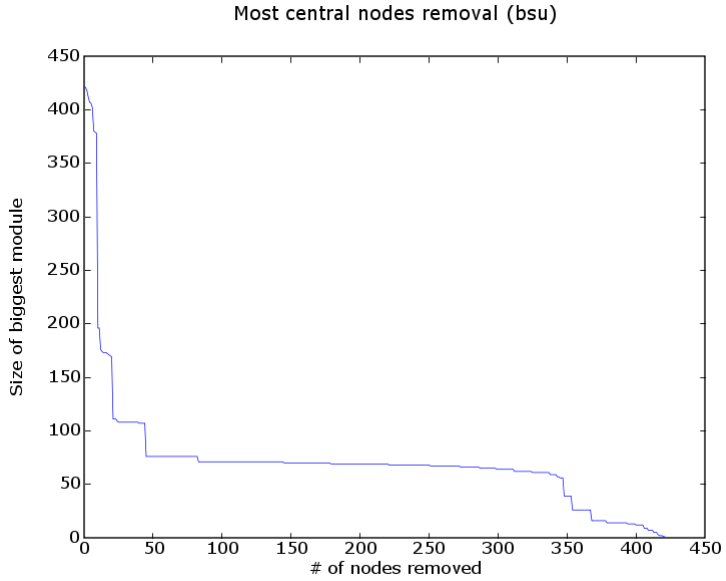
Biggest connected part change



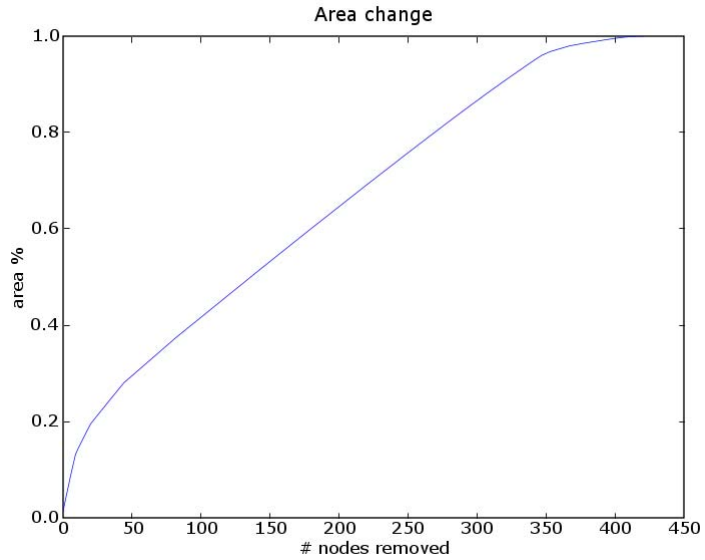
Biggest connected part change in *A. pernix* network.



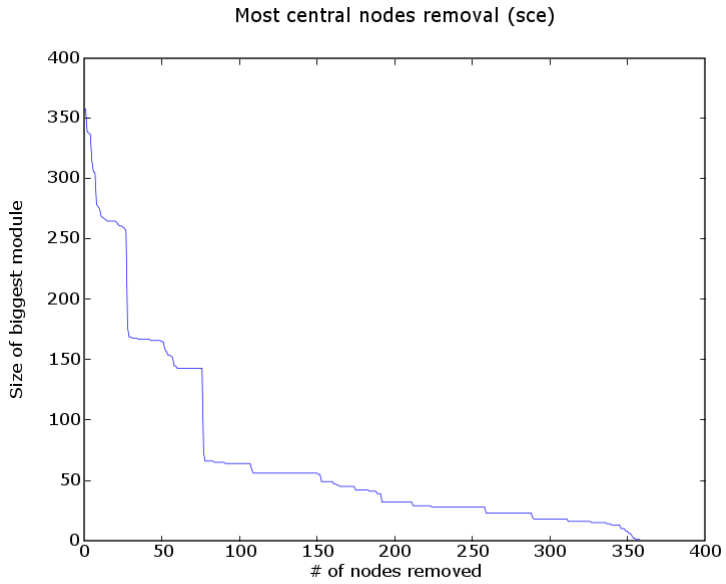
Biggest connected part change (integrated) in *A. pernix* network.



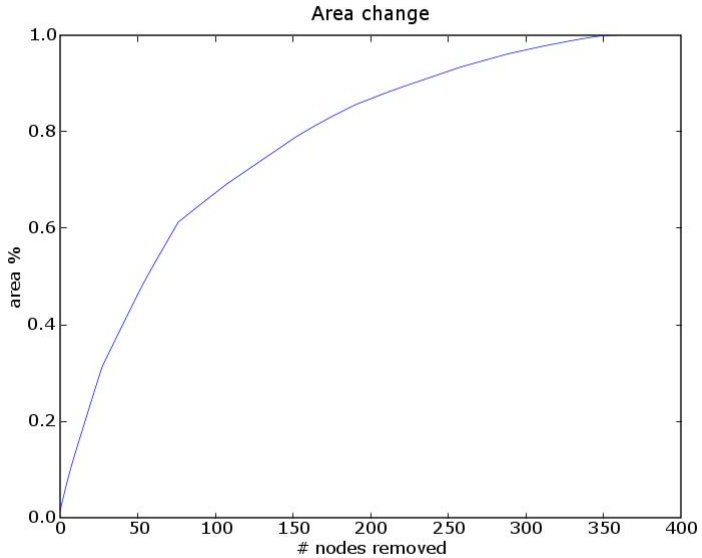
Biggest connected part change in *B. subtilis* network.



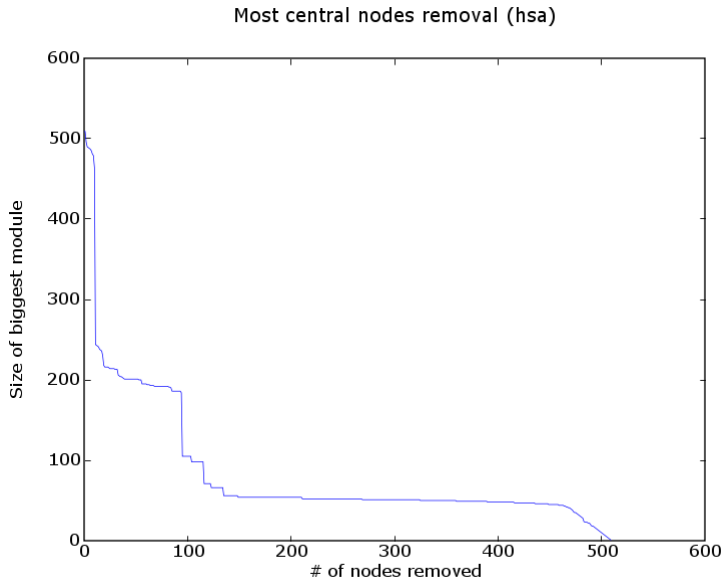
Biggest connected part change (integrated) in *B. subtilis* network.



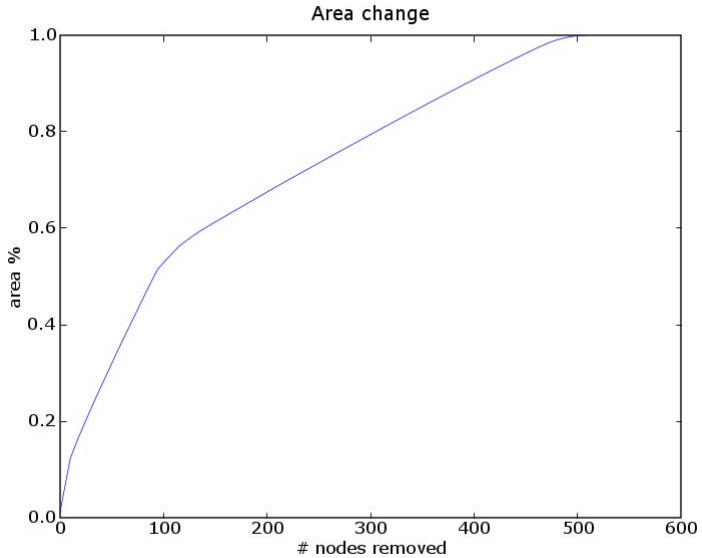
Biggest connected part change in *S. cerevisiae* network.



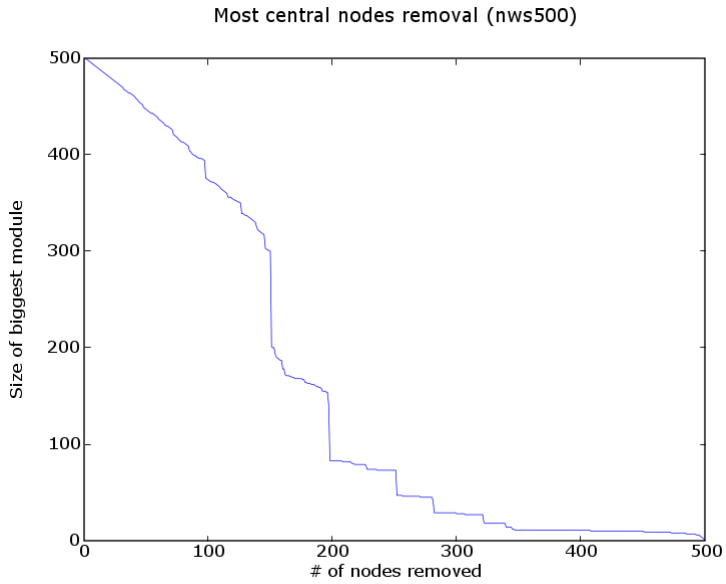
Biggest connected part change (integrated) in *S. cerevisiae* network.



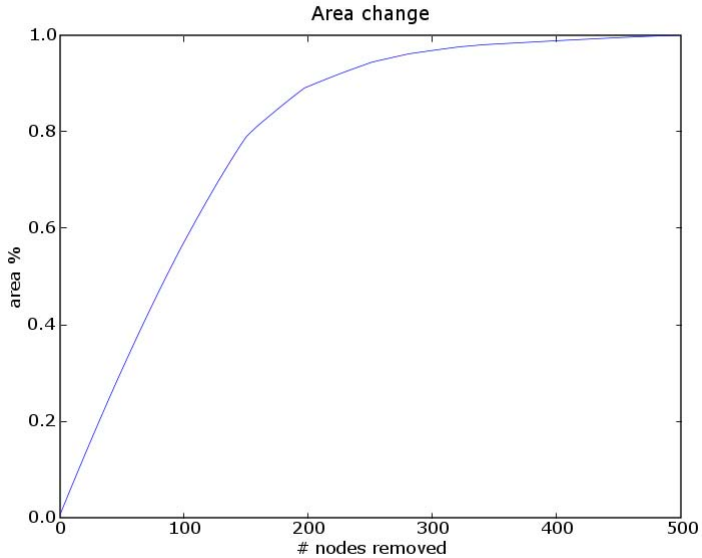
Biggest connected part change in *H. sapiens* network.



Biggest connected part change (integrated) in *H. sapiens* network.



Biggest connected part change in Newman-Watts-Strogatz network.



Biggest connected part change (integrated) in Newman-Watts-Strogatz network.

Appendix D
Modules distribution

Pathways per module

In section 4.3.3 the pathways per module for *E. coli* where summarized. Here the results for other organisms.

Pathways in the different modules obtained by the decomposition method proposed based on modularity for *A. pernix*.

Module	Pathways
12	Urea cycle and metabolism of amino groups Pyrimidine metabolism
11	Glycerophospholipid metabolism Fructose and mannose metabolism Glycerolipid metabolism Pentose and glucuronate interconversions Pentose phosphate pathway
10	Phenylalanine, tyrosine and tryptophan biosynthesis
9	Citrate cycle (TCA cycle) Reductive carboxylate cycle (CO2 fixation)
8	Pyrimidine metabolism
7	Glycine, serine and threonine metabolism Cysteine metabolism Glycolysis / Gluconeogenesis
6	Lysine biosynthesis Glycine, serine and threonine metabolism
5	Purine metabolism Neuroactive ligand-receptor interaction
4	Methionine metabolism Selenoamino acid metabolism Glycine, serine and threonine metabolism Sulfur metabolism
3	Fatty acid biosynthesis
2	Arginine and proline metabolism Glutamate metabolism
1	Carbon fixation Glycolysis / Gluconeogenesis

Pathways in the different modules obtained by the decomposition method proposed based on modularity for *B. subtilis.*

Module	Pathways
13	Pantothenate and CoA biosynthesis Valine, leucine and isoleucine biosynthesis Valine, leucine and isoleucine degradation
12	Butanoate metabolism
11	Galactose metabolism Nucleotide sugars metabolism Starch and sucrose metabolism Phosphotransferase system (PTS)
10	Purine metabolism
17	Fatty acid biosynthesis Flavonoid biosynthesis
9	Glycine, serine and threonine metabolism Methionine metabolism
15	Glycerophospholipid metabolism Glycerolipid metabolism Glycine, serine and threonine metabolism
8	Pyrimidine metabolism
7	Phenylalanine, tyrosine and tryptophan biosynthesis Phenylalanine metabolism
6	Purine metabolism Pentose and glucuronate interconversions Pentose phosphate pathway
5	Urea cycle and metabolism of amino groups Arginine and proline metabolism Histidine metabolism
4	Lysine biosynthesis Peptidoglycan biosynthesis Aminosugars metabolism
3	Arginine and proline metabolism Glyoxylate and dicarboxylate metabolism
2	Folate biosynthesis Riboflavin metabolism One carbon pool by folate
1	Pentose and glucuronate interconversions Pentose phosphate pathway
14	Alanine and aspartate metabolism Citrate cycle (TCA cycle) Reductive carboxylate cycle (CO2 fixation) Glyoxylate and dicarboxylate metabolism Glutamate metabolism Pyruvate metabolism
16	Glycerolipid metabolism

	Fructose and mannose metabolism Biosynthesis of steroids Carbon fixation Pentose phosphate pathway Glycerophospholipid metabolism Glycolysis / Gluconeogenesis Glycine, serine and threonine metabolism

Pathways in the different modules obtained by the decomposition method proposed based on modularity for *S. cerevisiae.*

Module	Pathways
12	Glycine, serine and threonine metabolism Fatty acid biosynthesis Benzoate degradation via hydroxylation Glycolysis / Gluconeogenesis Pyruvate metabolism
12	Fructose and mannose metabolism Aminosugars metabolism Phosphotransferase system (PTS) Carbon fixation Glycerolipid metabolism Galactose metabolism
11	Valine, leucine and isoleucine biosynthesis Pantothenate and CoA biosynthesis Valine, leucine and isoleucine degradation
10	Purine metabolism Histidine metabolism Pentose phosphate pathway
	Pentose and glucuronate interconversions
15	Galactose metabolism
	Starch and sucrose metabolism
9	Citrate cycle (TCA cycle) Glyoxylate and dicarboxylate metabolism Glutamate metabolism Lysine biosynthesis
	Reductive carboxylate cycle (CO2 fixation)
8	Glycerophospholipid metabolism Phosphatidylinositol signaling system Glycine, serine and threonine metabolism Glycerolipid metabolism
7	Cysteine metabolism Carbon fixation Glycolysis / Gluconeogenesis Pyruvate metabolism
6	Methionine metabolism
	Glycine, serine and threonine metabolism
5	Purine metabolism
4	Folate biosynthesis Purine metabolism
3	Phenylalanine, tyrosine and tryptophan biosynthesis
2	Biosynthesis of steroids
1	Pyrimidine metabolism
14	Urea cycle and metabolism of amino groups
	Arginine and proline metabolism

Pathways in the different modules obtained by the decomposition method proposed based on modularity for *H. sapiens.*

Module	Pathways
13	Purine metabolism Nicotinate and nicotinamide metabolism
12	Galactose metabolism Fructose and mannose metabolism
11	Fatty acid biosynthesis Fatty acid metabolism
10	Fatty acid metabolism Fatty acid elongation in mitochondria
17	Purine metabolism
9	Tyrosine metabolism
15	Biosynthesis of steroids
8	Glycine, serine and threonine metabolism Fatty acid biosynthesis Pyruvate metabolism
7	Glutathione metabolism Methionine metabolism Prostaglandin and leukotriene metabolism Cysteine metabolism Selenoamino acid metabolism
6	Porphyrin and chlorophyll metabolism Starch and sucrose metabolism Pentose and glucuronate interconversions
5	Aminosugars metabolism Alanine and aspartate metabolism Glutamate metabolism Reductive carboxylate cycle (CO2 fixation)
4	Carbon fixation Glycine, serine and threonine metabolism Glycerolipid metabolism Fructose and mannose metabolism Methane metabolism Glycolysis / Gluconeogenesis
3	Pyrimidine metabolism
2	Phosphatidylinositol signaling system Inositol phosphate metabolism Glycerophospholipid metabolism Glycerolipid metabolism
1	Arginine and proline metabolism Urea cycle and metabolism of amino groups
14	Prostaglandin and leukotriene metabolism Glycerophospholipid metabolism Neuroactive ligand-receptor interaction
16	Valine, leucine and isoleucine degradation Valine, leucine and isoleucine biosynthesis

Pathways per module for E. coli (full)

Complete pathways per module for E. coli (sections 4.3.3, 5.4.1.1, 5.4.1.2 and 0). As this data is too extense, only *E. coli* will be shown. Data from other organisms available under request.

Modularity method

===[module13]===(4)===
map00632 3 Benzoate degradation via CoA ligation
map00650 3 Butanoate metabolism
map00310 2 Lysine degradation
map00280 2 Valine, leucine and isoleucine degradation
map00072 2 Synthesis and degradation of ketone bodies
map00071 2 Fatty acid metabolism
map00640 2 Propanoate metabolism
map00380 2 Tryptophan metabolism
map00350 1 Tyrosine metabolism
map00643 1 Styrene degradation
map00620 1 Pyruvate metabolism
map02020 1 Two-component system - General
===[module12]===(42)===
map00040 17 Pentose and glucuronate interconversions
map00230 14 Purine metabolism
map00030 10 Pentose phosphate pathway
map00340 6 Histidine metabolism
map00500 5 Starch and sucrose metabolism
map00710 4 Carbon fixation
map00520 4 Nucleotide sugars metabolism
eco02011 3 ABC transporters - Organism-specific
map02010 3 ABC transporters - General
map00053 3 Ascorbate and aldarate metabolism
map00860 2 Porphyrin and chlorophyll metabolism
bha02011 2 ABC transporters - Organism-specific
map00680 2 Methane metabolism
bsu02011 2 ABC transporters - Organism-specific
map00271 1 Methionine metabolism
vch05110 1 Cholera - Infection
map00251 1 Glutamate metabolism
map02031 1 Bacterial chemotaxis - Organism-specific
map00630 1 Glyoxylate and dicarboxylate metabolism
map00730 1 Thiamine metabolism
map00562 1 Inositol phosphate metabolism
vch02011 1 ABC transporters - Organism-specific
hin02011 1 ABC transporters - Organism-specific
map00240 1 Pyrimidine metabolism
pae02011 1 ABC transporters - Organism-specific
===[module11]===(29)===
map00240 29 Pyrimidine metabolism
map00550 2 Peptidoglycan biosynthesis
hsa04080 2 Neuroactive ligand-receptor interaction
map00770 1 Pantothenate and CoA biosynthesis
map00252 1 Alanine and aspartate metabolism
map00410 1 beta-Alanine metabolism
===[module10]===(41)===
map00051 17 Fructose and mannose metabolism
map00040 7 Pentose and glucuronate interconversions
map00030 6 Pentose phosphate pathway
map00530 5 Aminosugars metabolism
bsu02060 4 Phosphotransferase system (PTS)
map00710 4 Carbon fixation
map00410 4 beta-Alanine metabolism
eco02060 4 Phosphotransferase system (PTS)
map00031 4 Inositol metabolism
map00620 4 Pyruvate metabolism
map00010 3 Glycolysis / Gluconeogenesis
map00640 3 Propanoate metabolism
map00252 2 Alanine and aspartate metabolism
map00251 2 Glutamate metabolism
map00750 2 Vitamin B6 metabolism
map00630 2 Glyoxylate and dicarboxylate metabolism
map00561 2 Glycerolipid metabolism
map00680 2 Methane metabolism
map00052 2 Galactose metabolism
map00240 2 Pyrimidine metabolism
map00625 1 Tetrachloroethene degradation
map00790 1 Folate biosynthesis
map00330 1 Arginine and proline metabolism
map00510 1 N-Glycan biosynthesis
map00100 1 Biosynthesis of steroids
map00400 1 Phenylalanine, tyrosine and tryptophan biosynthesis
map00770 1 Pantothenate and CoA biosynthesis
map00730 1 Thiamine metabolism
mpn02060 1 Phosphotransferase system (PTS)
map00564 1 Glycerophospholipid metabolism
map00540 1 Lipopolysaccharide biosynthesis
map00641 1 3-Chloroacrylic acid degradation
map00643 1 Styrene degradation
map00760 1 Nicotinate and nicotinamide metabolism
hsa04080 1 Neuroactive ligand-receptor interaction
===[module15]===(6)===
map00280 4 Valine, leucine and isoleucine degradation
map00640 3 Propanoate metabolism
map00660 2 C5-Branched dibasic acid metabolism
map00642 1 Ethylbenzene degradation
map00522 1 Biosynthesis of 12-, 14- and 16-membered macrolides
map01056 1 Biosynthesis of type II polyketide backbone
map00410 1 beta-Alanine metabolism
===[module9]===(27)===
map00230 23 Purine metabolism
hsa04080 3 Neuroactive ligand-receptor interaction
map00271 2 Methionine metabolism
hsa04020 2 Calcium signaling pathway
map00330 2 Arginine and proline metabolism
vch05110 2 Cholera - Infection
map00195 2 Photosynthesis
map00190 2 Oxidative phosphorylation
map00231 1 Puromycin biosynthesis
map00251 1 Glutamate metabolism
sce04110 1 Cell cycle
map00252 1 Alanine and aspartate metabolism
hsa04910 1 Insulin signaling pathway

hsa04010 1 MAPK signaling pathway
hsa04920 1 Adipocytokine signaling pathway
hsa04930 1 Type II diabetes mellitus
hsa04540 1 Gap junction
===[module8]===(6)===
map00770 6 Pantothenate and CoA biosynthesis
map00071 2 Fatty acid metabolism
map00230 1 Purine metabolism
map00540 1 Lipopolysaccharide biosynthesis
map00532 1 Chondroitin / Heparan sulfate biosynthesis
map00061 1 Fatty acid biosynthesis
map00020 1 Citrate cycle (TCA cycle)
map00920 1 Sulfur metabolism
===[module7]===(41)===
map00052 18 Galactose metabolism
map00520 12 Nucleotide sugars metabolism
map00500 9 Starch and sucrose metabolism
eco02060 8 Phosphotransferase system (PTS)
map00521 6 Streptomycin biosynthesis
bsu02060 5 Phosphotransferase system (PTS)
map00562 5 Inositol phosphate metabolism
map04070 4 Phosphatidylinositol signaling system
map00523 4 Polyketide sugar unit biosynthesis
map00010 4 Glycolysis / Gluconeogenesis
map00040 3 Pentose and glucuronate interconversions
map00051 3 Fructose and mannose metabolism
map00030 3 Pentose phosphate pathway
map02010 2 ABC transporters - General
map00561 2 Glycerolipid metabolism
map02020 2 Two-component system - General
eco02011 1 ABC transporters - Organism-specific
bbu02060 1 Phosphotransferase system (PTS)
map00350 1 Tyrosine metabolism
map02031 1 Bacterial chemotaxis - Organism-specific
mge02060 1 Phosphotransferase system (PTS)
map00901 1 Indole and ipecac alkaloid biosynthesis
hin02060 1 Phosphotransferase system (PTS)
eco02021 1 Two-component system - Organism-specific
hsa04020 1 Calcium signaling pathway
mpn02060 1 Phosphotransferase system (PTS)
map00363 1 Bisphenol A degradation
map00361 1 gamma-Hexachlorocyclohexane degradation
map00031 1 Inositol metabolism
hsa04930 1 Type II diabetes mellitus
map00740 1 Riboflavin metabolism
map00240 1 Pyrimidine metabolism
hin02011 1 ABC transporters - Organism-specific
hsa04910 1 Insulin signaling pathway
map01051 1 Biosynthesis of ansamycins
map01055 1 Biosynthesis of vancomycin group antibiotics
map00401 1 Novobiocin biosynthesis
===[module6]===(45)===
map00260 13 Glycine, serine and threonine metabolism
map00271 10 Methionine metabolism
map00480 9 Glutathione metabolism
map00450 7 Selenoamino acid metabolism
map00770 7 Pantothenate and CoA biosynthesis
map00920 7 Sulfur metabolism
map00272 6 Cysteine metabolism
map00970 5 Aminoacyl-tRNA synthetases
map00680 3 Methane metabolism
ana02011 2 ABC transporters - Organism-specific
map00330 2 Arginine and proline metabolism
map00251 2 Glutamate metabolism
syn02011 2 ABC transporters - Organism-specific
map00640 2 Propanoate metabolism
map00460 2 Cyanoamino acid metabolism
eco02011 1 ABC transporters - Organism-specific
map00310 1 Lysine degradation
map00311 1 Penicillins and cephalosporins biosynthesis
map00230 1 Purine metabolism
map00120 1 Bile acid biosynthesis
map00410 1 beta-Alanine metabolism
map00430 1 Taurine and hypotaurine metabolism
map02010 1 ABC transporters - General
map00860 1 Porphyrin and chlorophyll metabolism
map00220 1 Urea cycle and metabolism of amino groups
map00300 1 Lysine biosynthesis
map00660 1 C5-Branched dibasic acid metabolism
map00290 1 Valine, leucine and isoleucine biosynthesis
map00910 1 Nitrogen metabolism
map00620 1 Pyruvate metabolism
map00600 1 Glycosphingolipid metabolism
hsa04080 1 Neuroactive ligand-receptor interaction
===[module5]===(26)===
map00300 12 Lysine biosynthesis
map00550 9 Peptidoglycan biosynthesis
map00530 7 Aminosugars metabolism
map00310 3 Lysine degradation
map00471 3 D-Glutamine and D-glutamate metabolism
map00960 2 Alkaloid biosynthesis II
map00260 2 Glycine, serine and threonine metabolism
eco02011 1 ABC transporters - Organism-specific
map02010 1 ABC transporters - General
map00540 1 Lipopolysaccharide biosynthesis
map00780 1 Biotin metabolism
sce00563 1 Glycosylphosphatidylinositol(GPI)-anchor
map00970 1 Aminoacyl-tRNA synthetases
===[module4]===(34)===
map00790 12 Folate biosynthesis
map00740 9 Riboflavin metabolism
map00230 9 Purine metabolism
map00670 8 One carbon pool by folate
map00630 3 Glyoxylate and dicarboxylate metabolism
map00680 2 Methane metabolism
map00970 2 Aminoacyl-tRNA synthetases
map00260 2 Glycine, serine and threonine metabolism
hsa04540 1 Gap junction
map00623 1 2,4-Dichlorobenzoate degradation
vch05110 1 Cholera - Infection
===[module3]===(33)===
map00330 18 Arginine and proline metabolism
map00220 17 Urea cycle and metabolism of amino groups
map02010 8 ABC transporters - General
map00340 8 Histidine metabolism
eco02011 7 ABC transporters - Organism-specific
map00251 6 Glutamate metabolism
map00410 6 beta-Alanine metabolism
ana02011 5 ABC transporters - Organism-specific
map00970 5 Aminoacyl-tRNA synthetases
map00471 4 D-Glutamine and D-glutamate metabolism
pae02011 4 ABC transporters - Organism-specific
syn02011 3 ABC transporters - Organism-specific
hin02011 3 ABC transporters - Organism-specific
map00910 3 Nitrogen metabolism
map00230 2 Purine metabolism
map00252 2 Alanine and aspartate metabolism
map00650 2 Butanoate metabolism
vch02011 2 ABC transporters - Organism-specific
map00240 2 Pyrimidine metabolism
map00472 2 D-Arginine and D-ornithine metabolism
hsa04080 2 Neuroactive ligand-receptor interaction
map00331 1 Clavulanic acid biosynthesis
map00750 1 Vitamin B6 metabolism
hsa04540 1 Gap junction
map00960 1 Alkaloid biosynthesis II
map00860 1 Porphyrin and chlorophyll metabolism
map00480 1 Glutathione metabolism
bsu02011 1 ABC transporters - Organism-specific
map00660 1 C5-Branched dibasic acid metabolism

map00401 1 Novobiocin biosynthesis
===[module2]===(31)===
map00400 21 Phenylalanine, tyrosine and tryptophan biosynthesis
map00010 5 Glycolysis / Gluconeogenesis
map00710 3 Carbon fixation
map00360 3 Phenylalanine metabolism
map00260 3 Glycine, serine and threonine metabolism
map00401 3 Novobiocin biosynthesis
map00380 3 Tryptophan metabolism
map00970 3 Aminoacyl-tRNA synthetases
map00350 2 Tyrosine metabolism
map00540 2 Lipopolysaccharide biosynthesis
map00362 2 Benzoate degradation via hydroxylation
map00950 2 Alkaloid biosynthesis I
map01055 2 Biosynthesis of vancomycin group antibiotics
bsu02060 1 Phosphotransferase system (PTS)
map00790 1 Folate biosynthesis
ana02011 1 ABC transporters - Organism-specific
bbu02060 1 Phosphotransferase system (PTS)
map00440 1 Aminophosphonate metabolism
map00629 1 Carbazole degradation
map00020 1 Citrate cycle (TCA cycle)
mge02060 1 Phosphotransferase system (PTS)
map00630 1 Glyoxylate and dicarboxylate metabolism
map00632 1 Benzoate degradation via CoA ligation
map00960 1 Alkaloid biosynthesis II
hin02060 1 Phosphotransferase system (PTS)
mpn02060 1 Phosphotransferase system (PTS)
eco02060 1 Phosphotransferase system (PTS)
map00561 1 Glycerolipid metabolism
map00130 1 Ubiquinone biosynthesis
syn02011 1 ABC transporters - Organism-specific
map00901 1 Indole and ipecac alkaloid biosynthesis
map00720 1 Reductive carboxylate cycle (CO2 fixation)
map00623 1 2,4-Dichlorobenzoate degradation
map00620 1 Pyruvate metabolism
map01053 1 Biosynthesis of siderophore group nonribosomal
===[module1]===(34)===
map00564 13 Glycerophospholipid metabolism
map00561 8 Glycerolipid metabolism
map00330 6 Arginine and proline metabolism
map00630 6 Glyoxylate and dicarboxylate metabolism
map00053 6 Ascorbate and aldarate metabolism
map00260 6 Glycine, serine and threonine metabolism
map04070 3 Phosphatidylinositol signaling system
map00230 2 Purine metabolism
eco02011 1 ABC transporters - Organism-specific
bha02011 1 ABC transporters - Organism-specific
hsa00563 1 Glycosylphosphatidylinositol(GPI)-anchor
map00631 1 1,2-Dichloroethane degradation
map02010 1 ABC transporters - General
hsa04540 1 Gap junction
map00562 1 Inositol phosphate metabolism
map00051 1 Fructose and mannose metabolism
map00052 1 Galactose metabolism
map00680 1 Methane metabolism
map00361 1 gamma-Hexachlorocyclohexane degradation
map00660 1 C5-Branched dibasic acid metabolism
vch02011 1 ABC transporters - Organism-specific
map00627 1 1,4-Dichlorobenzene degradation
map00440 1 Aminophosphonate metabolism
sce00563 1 Glycosylphosphatidylinositol(GPI)-anchor
===[module14]===(39)===
map00020 14 Citrate cycle (TCA cycle)
map00720 11 Reductive carboxylate cycle (CO2 fixation)
map00252 9 Alanine and aspartate metabolism
map00630 9 Glyoxylate and dicarboxylate metabolism
map00251 7 Glutamate metabolism
map00640 7 Propanoate metabolism
map00010 7 Glycolysis / Gluconeogenesis
map00620 7 Pyruvate metabolism
map00061 6 Fatty acid biosynthesis
map00632 5 Benzoate degradation via CoA ligation
map00650 5 Butanoate metabolism
map00362 5 Benzoate degradation via hydroxylation
map00380 5 Tryptophan metabolism
map00310 4 Lysine degradation
map00300 4 Lysine biosynthesis
map00253 3 Tetracycline biosynthesis
map00350 3 Tyrosine metabolism
map00410 3 beta-Alanine metabolism
map00710 3 Carbon fixation
map00340 3 Histidine metabolism
map00760 3 Nicotinate and nicotinamide metabolism
map00360 3 Phenylalanine metabolism
map00280 3 Valine, leucine and isoleucine degradation
map00660 3 C5-Branched dibasic acid metabolism
map00430 3 Taurine and hypotaurine metabolism
map00625 2 Tetrachloroethene degradation
map00643 2 Styrene degradation
map00622 2 Toluene and xylene degradation
map00330 2 Arginine and proline metabolism
map00750 2 Vitamin B6 metabolism
map00190 2 Oxidative phosphorylation
map00220 2 Urea cycle and metabolism of amino groups
map00522 2 Biosynthesis of 12-, 14- and 16-membered
macrolides
map00071 2 Fatty acid metabolism
map00642 2 Ethylbenzene degradation
map00460 2 Cyanoamino acid metabolism
map00910 2 Nitrogen metabolism
map01056 2 Biosynthesis of type II polyketide backbone
map00970 2 Aminoacyl-tRNA synthetases
map00440 2 Aminophosphonate metabolism
eco02011 1 ABC transporters - Organism-specific
map01055 1 Biosynthesis of vancomycin group antibiotics
map00260 1 Glycine, serine and threonine metabolism
map00480 1 Glutathione metabolism
map00062 1 Fatty acid elongation in mitochondria
map00532 1 Chondroitin / Heparan sulfate biosynthesis
map02031 1 Bacterial chemotaxis - Organism-specific
map00920 1 Sulfur metabolism
map00031 1 Inositol metabolism
map00471 1 D-Glutamine and D-glutamate metabolism
map02010 1 ABC transporters - General
map00564 1 Glycerophospholipid metabolism
map00450 1 Selenoamino acid metabolism
map00053 1 Ascorbate and aldarate metabolism
map00680 1 Methane metabolism
map00361 1 gamma-Hexachlorocyclohexane degradation
map00072 1 Synthesis and degradation of ketone bodies
map00641 1 3-Chloroacrylic acid degradation
map00930 1 Caprolactam degradation
map00629 1 Carbazole degradation
map00628 1 Fluorene degradation
map00627 1 1,4-Dichlorobenzene degradation
map00624 1 1- and 2-Methylnaphthalene degradation
map00621 1 Biphenyl degradation
pae02011 1 ABC transporters - Organism-specific
hsa04080 1 Neuroactive ligand-receptor interaction
===[module16]===(35)===
map00290 13 Valine, leucine and isoleucine biosynthesis
map00100 6 Biosynthesis of steroids
map00280 6 Valine, leucine and isoleucine degradation
map00770 5 Pantothenate and CoA biosynthesis
ana02011 4 ABC transporters - Organism-specific
map00473 4 D-Alanine metabolism
map00970 4 Aminoacyl-tRNA synthetases
map00260 4 Glycine, serine and threonine metabolism
map00620 4 Pyruvate metabolism

eco02011 3 ABC transporters - Organism-specific
map00272 3 Cysteine metabolism
map00252 3 Alanine and aspartate metabolism
map02010 3 ABC transporters - General
map00052 3 Galactose metabolism
pae02011 3 ABC transporters - Organism-specific
map00430 2 Taurine and hypotaurine metabolism
map00550 2 Peptidoglycan biosynthesis
map00750 2 Vitamin B6 metabolism
map00710 2 Carbon fixation
syn02011 2 ABC transporters - Organism-specific
map00650 2 Butanoate metabolism
map00720 2 Reductive carboxylate cycle (CO2 fixation)
bsu02060 1 Phosphotransferase system (PTS)
map00311 1 Penicillins and cephalosporins biosynthesis
map00643 1 Styrene degradation
map00040 1 Pentose and glucuronate interconversions
map00331 1 Clavulanic acid biosynthesis
bbu02060 1 Phosphotransferase system (PTS)
map00350 1 Tyrosine metabolism
map00020 1 Citrate cycle (TCA cycle)
mge02060 1 Phosphotransferase system (PTS)
map00450 1 Selenoamino acid metabolism
map00960 1 Alkaloid biosynthesis II
hin02060 1 Phosphotransferase system (PTS)
eco02060 1 Phosphotransferase system (PTS)
mpn02060 1 Phosphotransferase system (PTS)
map00630 1 Glyoxylate and dicarboxylate metabolism
map00330 1 Arginine and proline metabolism
map00010 1 Glycolysis / Gluconeogenesis
map00360 1 Phenylalanine metabolism
map00053 1 Ascorbate and aldarate metabolism
map00640 1 Propanoate metabolism
map00660 1 C5-Branched dibasic acid metabolism
map00030 1 Pentose phosphate pathway
hsa04930 1 Type II diabetes mellitus
map00629 1 Carbazole degradation
map00642 1 Ethylbenzene degradation
map00628 1 Fluorene degradation
map00362 1 Benzoate degradation via hydroxylation
map00626 1 Nitrobenzene degradation
map00627 1 1,4-Dichlorobenzene degradation
map00624 1 1- and 2-Methylnaphthalene degradation
map00622 1 Toluene and xylene degradation
map00621 1 Biphenyl degradation
map00730 1 Thiamine metabolism

Degree centrality

===[module13]===(58)===
map00240 31 Pyrimidine metabolism
map00330 17 Arginine and proline metabolism
map00220 16 Urea cycle and metabolism of amino groups
map00410 7 beta-Alanine metabolism
map02010 7 ABC transporters - General
eco02011 6 ABC transporters - Organism-specific
map00340 6 Histidine metabolism
ana02011 4 ABC transporters - Organism-specific
map00251 4 Glutamate metabolism
pae02011 4 ABC transporters - Organism-specific
map00970 4 Aminoacyl-tRNA synthetases
map00252 3 Alanine and aspartate metabolism
hin02011 3 ABC transporters - Organism-specific
hsa04080 3 Neuroactive ligand-receptor interaction
map00230 2 Purine metabolism
map00472 2 D-Arginine and D-ornithine metabolism
map00550 2 Peptidoglycan biosynthesis
syn02011 2 ABC transporters - Organism-specific
vch02011 2 ABC transporters - Organism-specific
map00910 2 Nitrogen metabolism
map00331 1 Clavulanic acid biosynthesis
map00770 1 Pantothenate and CoA biosynthesis
map00650 1 Butanoate metabolism
map00471 1 D-Glutamine and D-glutamate metabolism
bsu02011 1 ABC transporters - Organism-specific
map00401 1 Novobiocin biosynthesis
map00960 1 Alkaloid biosynthesis II
===[module12]===(4)===
map00530 4 Aminosugars metabolism
map00251 2 Glutamate metabolism
bsu02060 2 Phosphotransferase system (PTS)
eco02060 1 Phosphotransferase system (PTS)
===[module11]===(3)===
map00260 3 Glycine, serine and threonine metabolism
===[module10]===(5)===
map00100 5 Biosynthesis of steroids
===[module17]===(3)===
map00052 3 Galactose metabolism
===[module16]===(5)===
map04070 4 Phosphatidylinositol signaling system
map00562 4 Inositol phosphate metabolism
map00521 2 Streptomycin biosynthesis
map00052 2 Galactose metabolism
map00031 1 Inositol metabolism
===[module15]===(3)===
map00030 3 Pentose phosphate pathway
map00240 1 Pyrimidine metabolism
===[module14]===(77)===
map00052 15 Galactose metabolism
map00480 11 Glutathione metabolism
map00230 10 Purine metabolism
map00260 9 Glycine, serine and threonine metabolism
eco02060 9 Phosphotransferase system (PTS)
map00030 9 Pentose phosphate pathway
map00630 8 Glyoxylate and dicarboxylate metabolism
map00010 8 Glycolysis / Gluconeogenesis
map00051 7 Fructose and mannose metabolism
map00620 7 Pyruvate metabolism
map00252 6 Alanine and aspartate metabolism
map00251 6 Glutamate metabolism
map00710 6 Carbon fixation
map00330 6 Arginine and proline metabolism
eco02011 5 ABC transporters - Organism-specific
map00970 5 Aminoacyl-tRNA synthetases
map00040 5 Pentose and glucuronate interconversions
map02010 5 ABC transporters - General
map00680 5 Methane metabolism
bsu02060 5 Phosphotransferase system (PTS)
map00340 5 Histidine metabolism
map00500 5 Starch and sucrose metabolism
map00650 4 Butanoate metabolism
map00660 4 C5-Branched dibasic acid metabolism
map00271 4 Methionine metabolism
map00272 4 Cysteine metabolism
map00471 4 D-Glutamine and D-glutamate metabolism
map00910 4 Nitrogen metabolism
map00020 4 Citrate cycle (TCA cycle)
map00720 4 Reductive carboxylate cycle (CO2 fixation)
map00300 3 Lysine biosynthesis
map00380 3 Tryptophan metabolism
map00460 3 Cyanoamino acid metabolism
hsa04080 3 Neuroactive ligand-receptor interaction
mpn02060 3 Phosphotransferase system (PTS)
map00362 3 Benzoate degradation via hydroxylation
map00760 3 Nicotinate and nicotinamide metabolism
map00750 3 Vitamin B6 metabolism
map02031 3 Bacterial chemotaxis - Organism-specific
map00220 3 Urea cycle and metabolism of amino groups
map00031 3 Inositol metabolism
map00430 3 Taurine and hypotaurine metabolism
map00310 2 Lysine degradation
map00253 2 Tetracycline biosynthesis
map00410 2 beta-Alanine metabolism
hin02011 2 ABC transporters - Organism-specific
map00770 2 Pantothenate and CoA biosynthesis
map00561 2 Glycerolipid metabolism
hsa04540 2 Gap junction
map00053 2 Ascorbate and aldarate metabolism
map00361 2 gamma-Hexachlorocyclohexane degradation
map02020 2 Two-component system - General
pae02011 2 ABC transporters - Organism-specific
bbu02060 2 Phosphotransferase system (PTS)
map00350 2 Tyrosine metabolism
map00100 2 Biosynthesis of steroids
hin02060 2 Phosphotransferase system (PTS)
syn02011 2 ABC transporters - Organism-specific
map00740 2 Riboflavin metabolism
map00627 2 1,4-Dichlorobenzene degradation
ana02011 2 ABC transporters - Organism-specific
vch05110 2 Cholera - Infection
mge02060 2 Phosphotransferase system (PTS)
map00730 2 Thiamine metabolism
hsa04930 2 Type II diabetes mellitus
map00642 2 Ethylbenzene degradation
map00311 1 Penicillins and cephalosporins biosynthesis
map00061 1 Fatty acid biosynthesis
map00062 1 Fatty acid elongation in mitochondria
map00860 1 Porphyrin and chlorophyll metabolism
bha02011 1 ABC transporters - Organism-specific
hsa04910 1 Insulin signaling pathway
map00072 1 Synthesis and degradation of ketone bodies
map00071 1 Fatty acid metabolism
map00790 1 Folate biosynthesis
map00473 1 D-Alanine metabolism
map00564 1 Glycerophospholipid metabolism
map00562 1 Inositol phosphate metabolism
map00363 1 Bisphenol A degradation
map00360 1 Phenylalanine metabolism
map00280 1 Valine, leucine and isoleucine degradation
map00290 1 Valine, leucine and isoleucine biosynthesis
map00901 1 Indole and ipecac alkaloid biosynthesis
eco02021 1 Two-component system - Organism-specific

hsa04020 1 Calcium signaling pathway
map00120 1 Bile acid biosynthesis
map00628 1 Fluorene degradation
map00629 1 Carbazole degradation
map00626 1 Nitrobenzene degradation
map00624 1 1- and 2-Methylnaphthalene degradation
map00625 1 Tetrachloroethene degradation
map00622 1 Toluene and xylene degradation
map00621 1 Biphenyl degradation
map01056 1 Biosynthesis of type II polyketide backbone
map00331 1 Clavulanic acid biosynthesis
map00530 1 Aminosugars metabolism
map00631 1 1,2-Dichloroethane degradation
map00632 1 Benzoate degradation via CoA ligation
map00920 1 Sulfur metabolism
map00521 1 Streptomycin biosynthesis
map00520 1 Nucleotide sugars metabolism
map00523 1 Polyketide sugar unit biosynthesis
map00522 1 Biosynthesis of 12-, 14- and 16-membered macrolides
bsu02011 1 ABC transporters - Organism-specific
vch02011 1 ABC transporters - Organism-specific
map00640 1 Propanoate metabolism
map00643 1 Styrene degradation
===[module19]===(28)===
map00230 28 Purine metabolism
hsa04080 3 Neuroactive ligand-receptor interaction
hsa04020 2 Calcium signaling pathway
vch05110 2 Cholera - Infection
map00195 2 Photosynthesis
map00190 2 Oxidative phosphorylation
map00231 1 Puromycin biosynthesis
hsa04910 1 Insulin signaling pathway
sce04110 1 Cell cycle
map00252 1 Alanine and aspartate metabolism
map00251 1 Glutamate metabolism
hsa04010 1 MAPK signaling pathway
hsa04920 1 Adipocytokine signaling pathway
hsa04930 1 Type II diabetes mellitus
hsa04540 1 Gap junction
===[module18]===(17)===
map00790 11 Folate biosynthesis
map00670 8 One carbon pool by folate
map00630 3 Glyoxylate and dicarboxylate metabolism
map00970 2 Aminoacyl-tRNA synthetases
map00260 2 Glycine, serine and threonine metabolism
map00680 2 Methane metabolism
map00623 1 2,4-Dichlorobenzoate degradation
===[module9]===(29)===
map00020 11 Citrate cycle (TCA cycle)
map00640 8 Propanoate metabolism
map00720 8 Reductive carboxylate cycle (CO2 fixation)
map00010 6 Glycolysis / Gluconeogenesis
map00280 6 Valine, leucine and isoleucine degradation
map00251 5 Glutamate metabolism
map00630 5 Glyoxylate and dicarboxylate metabolism
map00252 4 Alanine and aspartate metabolism
map00632 4 Benzoate degradation via CoA ligation
map00660 4 C5-Branched dibasic acid metabolism
map00620 4 Pyruvate metabolism
map00310 3 Lysine degradation
map00350 3 Tyrosine metabolism
map00650 3 Butanoate metabolism
map00360 3 Phenylalanine metabolism
map00362 3 Benzoate degradation via hydroxylation
map00430 2 Taurine and hypotaurine metabolism
map00440 2 Aminophosphonate metabolism
map00190 2 Oxidative phosphorylation
map00380 2 Tryptophan metabolism
map00642 2 Ethylbenzene degradation
map00643 2 Styrene degradation
map00622 2 Toluene and xylene degradation
map00625 1 Tetrachloroethene degradation
map00330 1 Arginine and proline metabolism
map00532 1 Chondroitin / Heparan sulfate biosynthesis
map00750 1 Vitamin B6 metabolism
map00410 1 beta-Alanine metabolism
map00920 1 Sulfur metabolism
map00710 1 Carbon fixation
map00564 1 Glycerophospholipid metabolism
map00220 1 Urea cycle and metabolism of amino groups
map00300 1 Lysine biosynthesis
map00450 1 Selenoamino acid metabolism
map00522 1 Biosynthesis of 12-, 14- and 16-membered macrolides
map00760 1 Nicotinate and nicotinamide metabolism
map00361 1 gamma-Hexachlorocyclohexane degradation
map00071 1 Fatty acid metabolism
map00641 1 3-Chloroacrylic acid degradation
map00930 1 Caprolactam degradation
map00629 1 Carbazole degradation
map00628 1 Fluorene degradation
map00624 1 1- and 2-Methylnaphthalene degradation
map00621 1 Biphenyl degradation
map01056 1 Biosynthesis of type II polyketide backbone
map00627 1 1,4-Dichlorobenzene degradation
===[module8]===(8)===
map00740 8 Riboflavin metabolism
===[module7]===(6)===
map00520 4 Nucleotide sugars metabolism
map00500 4 Starch and sucrose metabolism
map00052 3 Galactose metabolism
bsu02060 2 Phosphotransferase system (PTS)
map00040 2 Pentose and glucuronate interconversions
eco02060 2 Phosphotransferase system (PTS)
map00561 1 Glycerolipid metabolism
map01051 1 Biosynthesis of ansamycins
map00240 1 Pyrimidine metabolism
===[module6]===(5)===
map00040 4 Pentose and glucuronate interconversions
map00630 2 Glyoxylate and dicarboxylate metabolism
map00625 1 Tetrachloroethene degradation
map00750 1 Vitamin B6 metabolism
map00790 1 Folate biosynthesis
===[module5]===(9)===
map00051 7 Fructose and mannose metabolism
map00620 3 Pyruvate metabolism
map00561 1 Glycerolipid metabolism
map00643 1 Styrene degradation
map00010 1 Glycolysis / Gluconeogenesis
map00640 1 Propanoate metabolism
===[module4]===(5)===
map00330 5 Arginine and proline metabolism
===[module3]===(5)===
map00410 4 beta-Alanine metabolism
map00252 2 Alanine and aspartate metabolism
map00640 2 Propanoate metabolism
map00031 2 Inositol metabolism
map00770 1 Pantothenate and CoA biosynthesis
map00641 1 3-Chloroacrylic acid degradation
map00330 1 Arginine and proline metabolism
map00240 1 Pyrimidine metabolism
hsa04080 1 Neuroactive ligand-receptor interaction
===[module2]===(7)===
map00520 7 Nucleotide sugars metabolism
map00521 3 Streptomycin biosynthesis
map00523 3 Polyketide sugar unit biosynthesis
map00401 1 Novobiocin biosynthesis
map01055 1 Biosynthesis of vancomycin group antibiotics
===[module1]===(51)===

map00300 13 Lysine biosynthesis
map00550 9 Peptidoglycan biosynthesis
map00271 8 Methionine metabolism
map00260 8 Glycine, serine and threonine metabolism
map00530 7 Aminosugars metabolism
map00450 6 Selenoamino acid metabolism
map00920 4 Sulfur metabolism
map00290 4 Valine, leucine and isoleucine biosynthesis
map00970 4 Aminoacyl-tRNA synthetases
map00310 3 Lysine degradation
map00960 3 Alkaloid biosynthesis II
map00471 3 D-Glutamine and D-glutamate metabolism
map02010 3 ABC transporters - General
eco02011 2 ABC transporters - Organism-specific
map00640 2 Propanoate metabolism
map00280 2 Valine, leucine and isoleucine degradation
map00330 2 Arginine and proline metabolism
ana02011 1 ABC transporters - Organism-specific
map00860 1 Porphyrin and chlorophyll metabolism
map00780 1 Biotin metabolism
map00540 1 Lipopolysaccharide biosynthesis
map00660 1 C5-Branched dibasic acid metabolism
pae02011 1 ABC transporters - Organism-specific
sce00563 1 Glycosylphosphatidylinositol(GPI)-anchor
===[module26]===(23)===
map00040 17 Pentose and glucuronate interconversions
map00030 6 Pentose phosphate pathway
map00500 5 Starch and sucrose metabolism
map00520 4 Nucleotide sugars metabolism
map00053 3 Ascorbate and aldarate metabolism
map00710 3 Carbon fixation
map02010 2 ABC transporters - General
map00860 2 Porphyrin and chlorophyll metabolism
map00680 2 Methane metabolism
eco02011 2 ABC transporters - Organism-specific
map00630 1 Glyoxylate and dicarboxylate metabolism
bha02011 1 ABC transporters - Organism-specific
map00562 1 Inositol phosphate metabolism
bsu02011 1 ABC transporters - Organism-specific
===[module27]===(15)===
map00230 11 Purine metabolism
map00340 6 Histidine metabolism
map00730 1 Thiamine metabolism
map00240 1 Pyrimidine metabolism
map00251 1 Glutamate metabolism
map00030 1 Pentose phosphate pathway
===[module24]===(4)===
map00632 3 Benzoate degradation via CoA ligation
map00650 3 Butanoate metabolism
map00310 2 Lysine degradation
map00280 2 Valine, leucine and isoleucine degradation
map00072 2 Synthesis and degradation of ketone bodies
map00071 2 Fatty acid metabolism
map00640 2 Propanoate metabolism
map00380 2 Tryptophan metabolism
map00350 1 Tyrosine metabolism
map00643 1 Styrene degradation
map00620 1 Pyruvate metabolism
map02020 1 Two-component system - General
===[module25]===(7)===
map00061 5 Fatty acid biosynthesis
map00640 1 Propanoate metabolism
map00522 1 Biosynthesis of 12-, 14- and 16-membered macrolides
map00620 1 Pyruvate metabolism
map00253 1 Tetracycline biosynthesis
map01056 1 Biosynthesis of type II polyketide backbone
map01055 1 Biosynthesis of vancomycin group antibiotics
map00410 1 beta-Alanine metabolism
===[module22]===(65)===
map00400 22 Phenylalanine, tyrosine and tryptophan biosynthesis
map00564 13 Glycerophospholipid metabolism
map00561 9 Glycerolipid metabolism
map00260 9 Glycine, serine and threonine metabolism
map00053 6 Ascorbate and aldarate metabolism
map00010 5 Glycolysis / Gluconeogenesis
map00710 5 Carbon fixation
map00272 4 Cysteine metabolism
map00630 4 Glyoxylate and dicarboxylate metabolism
map00970 4 Aminoacyl-tRNA synthetases
map04070 3 Phosphatidylinositol signaling system
map00540 3 Lipopolysaccharide biosynthesis
map00360 3 Phenylalanine metabolism
map00401 3 Novobiocin biosynthesis
map00380 3 Tryptophan metabolism
map00350 2 Tyrosine metabolism
ana02011 2 ABC transporters - Organism-specific
map00920 2 Sulfur metabolism
map00051 2 Fructose and mannose metabolism
syn02011 2 ABC transporters - Organism-specific
map00680 2 Methane metabolism
map00362 2 Benzoate degradation via hydroxylation
map00440 2 Aminophosphonate metabolism
map00950 2 Alkaloid biosynthesis I
map01055 2 Biosynthesis of vancomycin group antibiotics
eco02011 1 ABC transporters - Organism-specific
bha02011 1 ABC transporters - Organism-specific
map00271 1 Methionine metabolism
map00030 1 Pentose phosphate pathway
bsu02060 1 Phosphotransferase system (PTS)
map00052 1 Galactose metabolism
bbu02060 1 Phosphotransferase system (PTS)
map00901 1 Indole and ipecac alkaloid biosynthesis
map00790 1 Folate biosynthesis
map00600 1 Glycosphingolipid metabolism
map00020 1 Citrate cycle (TCA cycle)
map00750 1 Vitamin B6 metabolism
hsa00563 1 Glycosylphosphatidylinositol(GPI)-anchor
map00632 1 Benzoate degradation via CoA ligation
map00960 1 Alkaloid biosynthesis II
map02010 1 ABC transporters - General
mge02060 1 Phosphotransferase system (PTS)
map00450 1 Selenoamino acid metabolism
hin02060 1 Phosphotransferase system (PTS)
map00620 1 Pyruvate metabolism
eco02060 1 Phosphotransferase system (PTS)
hsa04540 1 Gap junction
map00562 1 Inositol phosphate metabolism
map00130 1 Ubiquinone biosynthesis
vch02011 1 ABC transporters - Organism-specific
mpn02060 1 Phosphotransferase system (PTS)
map00629 1 Carbazole degradation
map00460 1 Cyanoamino acid metabolism
map00720 1 Reductive carboxylate cycle (CO2 fixation)
map00623 1 2,4-Dichlorobenzoate degradation
map01053 1 Biosynthesis of siderophore group nonribosomal
sce00563 1 Glycosylphosphatidylinositol(GPI)-anchor
===[module23]===(5)===
map00051 5 Fructose and mannose metabolism
map00510 1 N-Glycan biosynthesis
bsu02060 1 Phosphotransferase system (PTS)
eco02060 1 Phosphotransferase system (PTS)
===[module20]===(23)===
map00770 16 Pantothenate and CoA biosynthesis
map00290 9 Valine, leucine and isoleucine biosynthesis
map00280 4 Valine, leucine and isoleucine degradation
eco02011 2 ABC transporters - Organism-specific
ana02011 2 ABC transporters - Organism-specific
map02010 2 ABC transporters - General
map00970 2 Aminoacyl-tRNA synthetases

pae02011 2 ABC transporters - Organism-specific
map00071 2 Fatty acid metabolism
map00311 1 Penicillins and cephalosporins biosynthesis
map00230 1 Purine metabolism
map00650 1 Butanoate metabolism
map00640 1 Propanoate metabolism
map00620 1 Pyruvate metabolism
map00540 1 Lipopolysaccharide biosynthesis
syn02011 1 ABC transporters - Organism-specific
map00532 1 Chondroitin / Heparan sulfate biosynthesis
map00061 1 Fatty acid biosynthesis
map00020 1 Citrate cycle (TCA cycle)
map00410 1 beta-Alanine metabolism
map00920 1 Sulfur metabolism
===[module21]===(3)===
map00473 3 D-Alanine metabolism
map00550 2 Peptidoglycan biosynthesis
map00252 2 Alanine and aspartate metabolism
map00450 1 Selenoamino acid metabolism
map00720 1 Reductive carboxylate cycle (CO2 fixation)
ana02011 1 ABC transporters - Organism-specific
map00430 1 Taurine and hypotaurine metabolism
map00272 1 Cysteine metabolism
syn02011 1 ABC transporters - Organism-specific
map00710 1 Carbon fixation
map00970 1 Aminoacyl-tRNA synthetases
map00750 1 Vitamin B6 metabolism

Betweenness centrality

===[module13]===(46)===
map00230 15 Purine metabolism
map00030 8 Pentose phosphate pathway
map00710 7 Carbon fixation
map00340 7 Histidine metabolism
map00252 6 Alanine and aspartate metabolism
bsu02060 5 Phosphotransferase system (PTS)
map00530 5 Aminosugars metabolism
eco02060 5 Phosphotransferase system (PTS)
map00251 4 Glutamate metabolism
map00473 4 D-Alanine metabolism
map00720 4 Reductive carboxylate cycle (CO2 fixation)
map00620 4 Pyruvate metabolism
map00040 3 Pentose and glucuronate interconversions
map00272 3 Cysteine metabolism
map00400 3 Phenylalanine, tyrosine and tryptophan biosynthesis
map00020 3 Citrate cycle (TCA cycle)
map00750 3 Vitamin B6 metabolism
mpn02060 3 Phosphotransferase system (PTS)
map00630 3 Glyoxylate and dicarboxylate metabolism
map00260 3 Glycine, serine and threonine metabolism
map00970 3 Aminoacyl-tRNA synthetases
eco02011 2 ABC transporters - Organism-specific
map00910 2 Nitrogen metabolism
map00550 2 Peptidoglycan biosynthesis
bbu02060 2 Phosphotransferase system (PTS)
map00730 2 Thiamine metabolism
mge02060 2 Phosphotransferase system (PTS)
map00460 2 Cyanoamino acid metabolism
map02031 2 Bacterial chemotaxis - Organism-specific
map00430 2 Taurine and hypotaurine metabolism
map02010 2 ABC transporters - General
hin02060 2 Phosphotransferase system (PTS)
pae02011 2 ABC transporters - Organism-specific
map00051 2 Fructose and mannose metabolism
map00540 2 Lipopolysaccharide biosynthesis
map00010 2 Glycolysis / Gluconeogenesis
map00362 2 Benzoate degradation via hydroxylation
map00760 2 Nicotinate and nicotinamide metabolism
map00330 2 Arginine and proline metabolism
ana02011 1 ABC transporters - Organism-specific
map00643 1 Styrene degradation
map00271 1 Methionine metabolism
map00331 1 Clavulanic acid biosynthesis
map00440 1 Aminophosphonate metabolism
map00100 1 Biosynthesis of steroids
map00650 1 Butanoate metabolism
map00680 1 Methane metabolism
map00660 1 C5-Branched dibasic acid metabolism
map00300 1 Lysine biosynthesis
map00450 1 Selenoamino acid metabolism
map00350 1 Tyrosine metabolism
map00770 1 Pantothenate and CoA biosynthesis
map00410 1 beta-Alanine metabolism
bha02011 1 ABC transporters - Organism-specific
map00220 1 Urea cycle and metabolism of amino groups
vch05110 1 Cholera - Infection
syn02011 1 ABC transporters - Organism-specific
map00240 1 Pyrimidine metabolism
bsu02011 1 ABC transporters - Organism-specific
map00360 1 Phenylalanine metabolism
map00053 1 Ascorbate and aldarate metabolism
hsa04930 1 Type II diabetes mellitus
map00380 1 Tryptophan metabolism
vch02011 1 ABC transporters - Organism-specific
hin02011 1 ABC transporters - Organism-specific
map00642 1 Ethylbenzene degradation
map00628 1 Fluorene degradation
map00626 1 Nitrobenzene degradation
map00627 1 1,4-Dichlorobenzene degradation
map00624 1 1- and 2-Methylnaphthalene degradation
map00622 1 Toluene and xylene degradation
map00621 1 Biphenyl degradation
map00290 1 Valine, leucine and isoleucine biosynthesis
map00629 1 Carbazole degradation
map00253 1 Tetracycline biosynthesis
hsa04080 1 Neuroactive ligand-receptor interaction
===[module12]===(5)===
map00051 5 Fructose and mannose metabolism
map00510 1 N-Glycan biosynthesis
bsu02060 1 Phosphotransferase system (PTS)
eco02060 1 Phosphotransferase system (PTS)
===[module11]===(24)===
map00051 8 Fructose and mannose metabolism
map00040 5 Pentose and glucuronate interconversions
map00620 5 Pyruvate metabolism
map00410 4 beta-Alanine metabolism
map00031 3 Inositol metabolism
map00640 3 Propanoate metabolism
map00252 2 Alanine and aspartate metabolism
map00630 2 Glyoxylate and dicarboxylate metabolism
map00710 2 Carbon fixation
map00561 2 Glycerolipid metabolism
map00010 2 Glycolysis / Gluconeogenesis
map00260 2 Glycine, serine and threonine metabolism
map00625 1 Tetrachloroethene degradation
map00330 1 Arginine and proline metabolism
map00790 1 Folate biosynthesis
map00750 1 Vitamin B6 metabolism
map00770 1 Pantothenate and CoA biosynthesis
map00564 1 Glycerophospholipid metabolism
map00052 1 Galactose metabolism
map00240 1 Pyrimidine metabolism
map00540 1 Lipopolysaccharide biosynthesis
map00641 1 3-Chloroacrylic acid degradation
map00643 1 Styrene degradation
map00760 1 Nicotinate and nicotinamide metabolism
hsa04080 1 Neuroactive ligand-receptor interaction
===[module10]===(35)===
map00564 13 Glycerophospholipid metabolism
map00561 9 Glycerolipid metabolism
map00260 9 Glycine, serine and threonine metabolism
map00053 6 Ascorbate and aldarate metabolism
map00272 4 Cysteine metabolism
map00630 4 Glyoxylate and dicarboxylate metabolism
map04070 3 Phosphatidylinositol signaling system
map00920 2 Sulfur metabolism
map00051 2 Fructose and mannose metabolism
map00680 2 Methane metabolism
map00010 2 Glycolysis / Gluconeogenesis
eco02011 1 ABC transporters - Organism-specific
map00562 1 Inositol phosphate metabolism
ana02011 1 ABC transporters - Organism-specific
bha02011 1 ABC transporters - Organism-specific
map00271 1 Methionine metabolism
hsa00563 1 Glycosylphosphatidylinositol(GPI)-anchor
map02010 1 ABC transporters - General
map00710 1 Carbon fixation
hsa04540 1 Gap junction
map00450 1 Selenoamino acid metabolism
map00052 1 Galactose metabolism
syn02011 1 ABC transporters - Organism-specific
vch02011 1 ABC transporters - Organism-specific

map00460 1 Cyanoamino acid metabolism
map00440 1 Aminophosphonate metabolism
map00600 1 Glycosphingolipid metabolism
map00970 1 Aminoacyl-tRNA synthetases
sce00563 1 Glycosylphosphatidylinositol(GPI)-anchor
===[module15]===(60)===
map00230 31 Purine metabolism
map00790 12 Folate biosynthesis
map00740 9 Riboflavin metabolism
map00670 8 One carbon pool by folate
vch05110 3 Cholera - Infection
map00630 3 Glyoxylate and dicarboxylate metabolism
hsa04080 3 Neuroactive ligand-receptor interaction
map00271 2 Methionine metabolism
hsa04020 2 Calcium signaling pathway
map00330 2 Arginine and proline metabolism
map00195 2 Photosynthesis
map00190 2 Oxidative phosphorylation
hsa04540 2 Gap junction
map00680 2 Methane metabolism
map00260 2 Glycine, serine and threonine metabolism
map00970 2 Aminoacyl-tRNA synthetases
map00231 1 Puromycin biosynthesis
map00252 1 Alanine and aspartate metabolism
map00251 1 Glutamate metabolism
hsa04920 1 Adipocytokine signaling pathway
sce04110 1 Cell cycle
hsa04910 1 Insulin signaling pathway
hsa04010 1 MAPK signaling pathway
hsa04930 1 Type II diabetes mellitus
map00623 1 2,4-Dichlorobenzoate degradation
===[module9]===(29)===
map00240 29 Pyrimidine metabolism
map00550 2 Peptidoglycan biosynthesis
hsa04080 2 Neuroactive ligand-receptor interaction
map00770 1 Pantothenate and CoA biosynthesis
map00252 1 Alanine and aspartate metabolism
map00410 1 beta-Alanine metabolism
===[module8]===(31)===
map00480 8 Glutathione metabolism
map00260 7 Glycine, serine and threonine metabolism
map00640 6 Propanoate metabolism
map00061 6 Fatty acid biosynthesis
map00632 5 Benzoate degradation via CoA ligation
map00280 5 Valine, leucine and isoleucine degradation
map00310 4 Lysine degradation
map00650 4 Butanoate metabolism
map00620 4 Pyruvate metabolism
map00071 3 Fatty acid metabolism
map00680 3 Methane metabolism
map00072 3 Synthesis and degradation of ketone bodies
map00380 3 Tryptophan metabolism
map00362 3 Benzoate degradation via hydroxylation
map00253 2 Tetracycline biosynthesis
map00020 2 Citrate cycle (TCA cycle)
map00410 2 beta-Alanine metabolism
map00720 2 Reductive carboxylate cycle (CO2 fixation)
map00522 2 Biosynthesis of 12-, 14- and 16-membered macrolides
map01056 2 Biosynthesis of type II polyketide backbone
map00970 2 Aminoacyl-tRNA synthetases
map00625 1 Tetrachloroethene degradation
ana02011 1 ABC transporters - Organism-specific
map00272 1 Cysteine metabolism
map00120 1 Bile acid biosynthesis
map00252 1 Alanine and aspartate metabolism
map00251 1 Glutamate metabolism
map02020 1 Two-component system - General
map00350 1 Tyrosine metabolism
map00230 1 Purine metabolism
map00630 1 Glyoxylate and dicarboxylate metabolism
map00062 1 Fatty acid elongation in mitochondria
map02010 1 ABC transporters - General
map00860 1 Porphyrin and chlorophyll metabolism
map00220 1 Urea cycle and metabolism of amino groups
map00300 1 Lysine biosynthesis
syn02011 1 ABC transporters - Organism-specific
map00010 1 Glycolysis / Gluconeogenesis
map00031 1 Inositol metabolism
map00930 1 Caprolactam degradation
map00642 1 Ethylbenzene degradation
map00643 1 Styrene degradation
map00460 1 Cyanoamino acid metabolism
map00624 1 1- and 2-Methylnaphthalene degradation
map00430 1 Taurine and hypotaurine metabolism
map00910 1 Nitrogen metabolism
map01055 1 Biosynthesis of vancomycin group antibiotics
hsa04080 1 Neuroactive ligand-receptor interaction
===[module7]===(41)===
map00052 18 Galactose metabolism
map00520 12 Nucleotide sugars metabolism
map00500 9 Starch and sucrose metabolism
eco02060 8 Phosphotransferase system (PTS)
map00521 6 Streptomycin biosynthesis
bsu02060 5 Phosphotransferase system (PTS)
map00562 5 Inositol phosphate metabolism
map04070 4 Phosphatidylinositol signaling system
map00523 4 Polyketide sugar unit biosynthesis
map00010 4 Glycolysis / Gluconeogenesis
map00040 3 Pentose and glucuronate interconversions
map00030 3 Pentose phosphate pathway
map00051 3 Fructose and mannose metabolism
map02010 2 ABC transporters - General
map00561 2 Glycerolipid metabolism
map02020 2 Two-component system - General
eco02011 1 ABC transporters - Organism-specific
bbu02060 1 Phosphotransferase system (PTS)
map00350 1 Tyrosine metabolism
map02031 1 Bacterial chemotaxis - Organism-specific
mge02060 1 Phosphotransferase system (PTS)
map00901 1 Indole and ipecac alkaloid biosynthesis
hin02060 1 Phosphotransferase system (PTS)
eco02021 1 Two-component system - Organism-specific
hsa04020 1 Calcium signaling pathway
mpn02060 1 Phosphotransferase system (PTS)
map00363 1 Bisphenol A degradation
map00361 1 gamma-Hexachlorocyclohexane degradation
map00031 1 Inositol metabolism
map00401 1 Novobiocin biosynthesis
hsa04930 1 Type II diabetes mellitus
map00740 1 Riboflavin metabolism
map00240 1 Pyrimidine metabolism
hin02011 1 ABC transporters - Organism-specific
hsa04910 1 Insulin signaling pathway
map01051 1 Biosynthesis of ansamycins
map01055 1 Biosynthesis of vancomycin group antibiotics
===[module6]===(26)===
map00040 9 Pentose and glucuronate interconversions
map00030 8 Pentose phosphate pathway
map00100 6 Biosynthesis of steroids
map00710 5 Carbon fixation
map00052 4 Galactose metabolism
map00520 3 Nucleotide sugars metabolism
map00680 3 Methane metabolism
map00010 3 Glycolysis / Gluconeogenesis
map02010 2 ABC transporters - General
map00500 2 Starch and sucrose metabolism
eco02011 2 ABC transporters - Organism-specific
map00630 1 Glyoxylate and dicarboxylate metabolism
map00240 1 Pyrimidine metabolism

bha02011 1 ABC transporters - Organism-specific
map00730 1 Thiamine metabolism
map00051 1 Fructose and mannose metabolism
map00053 1 Ascorbate and aldarate metabolism
bsu02011 1 ABC transporters - Organism-specific
map00031 1 Inositol metabolism
===[module5]===(49)===
map00300 13 Lysine biosynthesis
map00550 9 Peptidoglycan biosynthesis
map00271 8 Methionine metabolism
map00530 7 Aminosugars metabolism
map00450 6 Selenoamino acid metabolism
map00260 6 Glycine, serine and threonine metabolism
map00920 4 Sulfur metabolism
map00290 4 Valine, leucine and isoleucine biosynthesis
map00310 3 Lysine degradation
map00960 3 Alkaloid biosynthesis II
map00471 3 D-Glutamine and D-glutamate metabolism
map00970 3 Aminoacyl-tRNA synthetases
eco02011 2 ABC transporters - Organism-specific
map02010 2 ABC transporters - General
map00640 2 Propanoate metabolism
map00280 2 Valine, leucine and isoleucine degradation
map00330 2 Arginine and proline metabolism
ana02011 1 ABC transporters - Organism-specific
map00780 1 Biotin metabolism
map00540 1 Lipopolysaccharide biosynthesis
map00660 1 C5-Branched dibasic acid metabolism
pae02011 1 ABC transporters - Organism-specific
sce00563 1 Glycosylphosphatidylinositol(GPI)-anchor
===[module4]===(10)===
map00040 8 Pentose and glucuronate interconversions
map00500 3 Starch and sucrose metabolism
map00860 2 Porphyrin and chlorophyll metabolism
map00053 2 Ascorbate and aldarate metabolism
map00562 1 Inositol phosphate metabolism
map00520 1 Nucleotide sugars metabolism
map00030 1 Pentose phosphate pathway
===[module3]===(25)===
map00770 17 Pantothenate and CoA biosynthesis
map00290 9 Valine, leucine and isoleucine biosynthesis
map00280 4 Valine, leucine and isoleucine degradation
eco02011 3 ABC transporters - Organism-specific
map00970 3 Aminoacyl-tRNA synthetases
map00311 2 Penicillins and cephalosporins biosynthesis
ana02011 2 ABC transporters - Organism-specific
map00920 2 Sulfur metabolism
map02010 2 ABC transporters - General
map00071 2 Fatty acid metabolism
map00480 2 Glutathione metabolism
pae02011 2 ABC transporters - Organism-specific
map00230 1 Purine metabolism
map00271 1 Methionine metabolism
map00272 1 Cysteine metabolism
map00251 1 Glutamate metabolism
map00532 1 Chondroitin / Heparan sulfate biosynthesis
map00061 1 Fatty acid biosynthesis
map00020 1 Citrate cycle (TCA cycle)
map00650 1 Butanoate metabolism
map00410 1 beta-Alanine metabolism
map00430 1 Taurine and hypotaurine metabolism
syn02011 1 ABC transporters - Organism-specific
map00540 1 Lipopolysaccharide biosynthesis
map00260 1 Glycine, serine and threonine metabolism
map00640 1 Propanoate metabolism
map00620 1 Pyruvate metabolism
===[module2]===(38)===
map00020 11 Citrate cycle (TCA cycle)
map00630 9 Glyoxylate and dicarboxylate metabolism
map00720 8 Reductive carboxylate cycle (CO2 fixation)
map00330 7 Arginine and proline metabolism
map00251 6 Glutamate metabolism
map00010 6 Glycolysis / Gluconeogenesis
map00660 6 C5-Branched dibasic acid metabolism
map00640 6 Propanoate metabolism
map00252 5 Alanine and aspartate metabolism
map00650 4 Butanoate metabolism
map00280 4 Valine, leucine and isoleucine degradation
map00620 4 Pyruvate metabolism
map00310 3 Lysine degradation
map00350 3 Tyrosine metabolism
map00632 3 Benzoate degradation via CoA ligation
map00380 3 Tryptophan metabolism
map00360 3 Phenylalanine metabolism
map00230 2 Purine metabolism
map00642 2 Ethylbenzene degradation
map00440 2 Aminophosphonate metabolism
map00750 2 Vitamin B6 metabolism
map00190 2 Oxidative phosphorylation
map00300 2 Lysine biosynthesis
map00340 2 Histidine metabolism
map00361 2 gamma-Hexachlorocyclohexane degradation
map00430 2 Taurine and hypotaurine metabolism
map00643 2 Styrene degradation
map00627 2 1,4-Dichlorobenzene degradation
map00622 2 Toluene and xylene degradation
map00625 1 Tetrachloroethene degradation
map00362 1 Benzoate degradation via hydroxylation
map00532 1 Chondroitin / Heparan sulfate biosynthesis
map00410 1 beta-Alanine metabolism
map00631 1 1,2-Dichloroethane degradation
map00920 1 Sulfur metabolism
map00471 1 D-Glutamine and D-glutamate metabolism
map00710 1 Carbon fixation
map00071 1 Fatty acid metabolism
map00564 1 Glycerophospholipid metabolism
map00220 1 Urea cycle and metabolism of amino groups
map00450 1 Selenoamino acid metabolism
map00522 1 Biosynthesis of 12-, 14- and 16-membered
macrolides
map00053 1 Ascorbate and aldarate metabolism
map00760 1 Nicotinate and nicotinamide metabolism
map00260 1 Glycine, serine and threonine metabolism
map00629 1 Carbazole degradation
map00641 1 3-Chloroacrylic acid degradation
map00628 1 Fluorene degradation
map00621 1 Biphenyl degradation
map01056 1 Biosynthesis of type II polyketide backbone
===[module1]===(21)===
map00400 19 Phenylalanine, tyrosine and tryptophan biosynthesis
map00360 3 Phenylalanine metabolism
map00970 3 Aminoacyl-tRNA synthetases
map00401 3 Novobiocin biosynthesis
map00380 3 Tryptophan metabolism
map00350 2 Tyrosine metabolism
map00362 2 Benzoate degradation via hydroxylation
map00950 2 Alkaloid biosynthesis I
map01055 2 Biosynthesis of vancomycin group antibiotics
map00790 1 Folate biosynthesis
map00632 1 Benzoate degradation via CoA ligation
ana02011 1 ABC transporters - Organism-specific
map00960 1 Alkaloid biosynthesis II
map00130 1 Ubiquinone biosynthesis
map00629 1 Carbazole degradation
map00623 1 2,4-Dichlorobenzoate degradation
syn02011 1 ABC transporters - Organism-specific
map01053 1 Biosynthesis of siderophore group nonribosomal
map00901 1 Indole and ipecac alkaloid biosynthesis
===[module14]===(33)===
map00330 18 Arginine and proline metabolism

map00220 17 Urea cycle and metabolism of amino groups
map02010 8 ABC transporters - General
map00340 8 Histidine metabolism
eco02011 7 ABC transporters - Organism-specific
map00251 6 Glutamate metabolism
map00410 6 beta-Alanine metabolism
ana02011 5 ABC transporters - Organism-specific
map00970 5 Aminoacyl-tRNA synthetases
map00471 4 D-Glutamine and D-glutamate metabolism
pae02011 4 ABC transporters - Organism-specific
syn02011 3 ABC transporters - Organism-specific
hin02011 3 ABC transporters - Organism-specific
map00910 3 Nitrogen metabolism
map00230 2 Purine metabolism
map00252 2 Alanine and aspartate metabolism
map00650 2 Butanoate metabolism
map00240 2 Pyrimidine metabolism
vch02011 2 ABC transporters - Organism-specific
map00472 2 D-Arginine and D-ornithine metabolism
hsa04080 2 Neuroactive ligand-receptor interaction
map00331 1 Clavulanic acid biosynthesis
map00750 1 Vitamin B6 metabolism
hsa04540 1 Gap junction
map00960 1 Alkaloid biosynthesis II
map00860 1 Porphyrin and chlorophyll metabolism
bsu02011 1 ABC transporters - Organism-specific
map00660 1 C5-Branched dibasic acid metabolism
map00401 1 Novobiocin biosynthesis
map00480 1 Glutathione metabolism

Closeness centrality

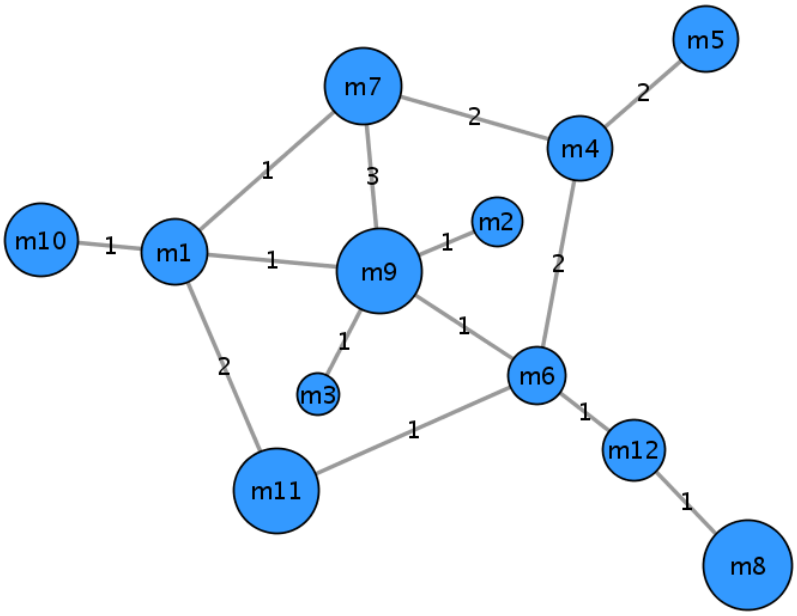
===[module13]===(5)===
map00280 2 Valine, leucine and isoleucine degradation
map00290 2 Valine, leucine and isoleucine biosynthesis
eco02011 1 ABC transporters - Organism-specific
pae02011 1 ABC transporters - Organism-specific
ana02011 1 ABC transporters - Organism-specific
map00960 1 Alkaloid biosynthesis II
map02010 1 ABC transporters - General
map00970 1 Aminoacyl-tRNA synthetases
===[module12]===(4)===
map00530 4 Aminosugars metabolism
map00251 2 Glutamate metabolism
bsu02060 2 Phosphotransferase system (PTS)
eco02060 1 Phosphotransferase system (PTS)
===[module11]===(41)===
map00052 18 Galactose metabolism
map00520 12 Nucleotide sugars metabolism
map00500 9 Starch and sucrose metabolism
eco02060 8 Phosphotransferase system (PTS)
map00521 6 Streptomycin biosynthesis
bsu02060 5 Phosphotransferase system (PTS)
map00562 5 Inositol phosphate metabolism
map04070 4 Phosphatidylinositol signaling system
map00523 4 Polyketide sugar unit biosynthesis
map00010 4 Glycolysis / Gluconeogenesis
map00040 3 Pentose and glucuronate interconversions
map00030 3 Pentose phosphate pathway
map00051 3 Fructose and mannose metabolism
map02010 2 ABC transporters - General
map00561 2 Glycerolipid metabolism
map02020 2 Two-component system - General
eco02011 1 ABC transporters - Organism-specific
bbu02060 1 Phosphotransferase system (PTS)
map00350 1 Tyrosine metabolism
map02031 1 Bacterial chemotaxis - Organism-specific
mge02060 1 Phosphotransferase system (PTS)
map00901 1 Indole and ipecac alkaloid biosynthesis
hin02060 1 Phosphotransferase system (PTS)
eco02021 1 Two-component system - Organism-specific
hsa04020 1 Calcium signaling pathway
mpn02060 1 Phosphotransferase system (PTS)
map00363 1 Bisphenol A degradation
map00361 1 gamma-Hexachlorocyclohexane degradation
map00031 1 Inositol metabolism
map00401 1 Novobiocin biosynthesis
hsa04930 1 Type II diabetes mellitus
map00740 1 Riboflavin metabolism
map00240 1 Pyrimidine metabolism
hin02011 1 ABC transporters - Organism-specific
hsa04910 1 Insulin signaling pathway
map01051 1 Biosynthesis of ansamycins
map01055 1 Biosynthesis of vancomycin group antibiotics
===[module10]===(49)===
map00230 41 Purine metabolism
map00340 6 Histidine metabolism
hsa04080 3 Neuroactive ligand-receptor interaction
hsa04020 2 Calcium signaling pathway
map00330 2 Arginine and proline metabolism
vch05110 2 Cholera - Infection
map00271 2 Methionine metabolism
map00251 2 Glutamate metabolism
map00195 2 Photosynthesis
map00190 2 Oxidative phosphorylation
map00231 1 Puromycin biosynthesis
map00252 1 Alanine and aspartate metabolism
map00730 1 Thiamine metabolism
sce04110 1 Cell cycle
map00240 1 Pyrimidine metabolism
hsa04910 1 Insulin signaling pathway
hsa04920 1 Adipocytokine signaling pathway
hsa04010 1 MAPK signaling pathway
map00030 1 Pentose phosphate pathway
hsa04930 1 Type II diabetes mellitus
hsa04540 1 Gap junction
===[module17]===(3)===
map00030 3 Pentose phosphate pathway
map00240 1 Pyrimidine metabolism
===[module16]===(23)===
map00340 8 Histidine metabolism
map00020 7 Citrate cycle (TCA cycle)
map00251 5 Glutamate metabolism
map00650 5 Butanoate metabolism
map00252 4 Alanine and aspartate metabolism
map00632 4 Benzoate degradation via CoA ligation
map00720 4 Reductive carboxylate cycle (CO2 fixation)
map00310 3 Lysine degradation
map00330 3 Arginine and proline metabolism
map00350 3 Tyrosine metabolism
map00410 3 beta-Alanine metabolism
map00640 3 Propanoate metabolism
map00380 3 Tryptophan metabolism
map00750 2 Vitamin B6 metabolism
map00630 2 Glyoxylate and dicarboxylate metabolism
map00190 2 Oxidative phosphorylation
map00300 2 Lysine biosynthesis
map00362 2 Benzoate degradation via hydroxylation
map00360 2 Phenylalanine metabolism
map00280 2 Valine, leucine and isoleucine degradation
map00010 2 Glycolysis / Gluconeogenesis
eco02011 1 ABC transporters - Organism-specific
ana02011 1 ABC transporters - Organism-specific
map00643 1 Styrene degradation
map00471 1 D-Glutamine and D-glutamate metabolism
map02010 1 ABC transporters - General
map00220 1 Urea cycle and metabolism of amino groups
map00053 1 Ascorbate and aldarate metabolism
map00760 1 Nicotinate and nicotinamide metabolism
map00361 1 gamma-Hexachlorocyclohexane degradation
syn02011 1 ABC transporters - Organism-specific
map00071 1 Fatty acid metabolism
map00660 1 C5-Branched dibasic acid metabolism
map00930 1 Caprolactam degradation
map00624 1 1- and 2-Methylnaphthalene degradation
pae02011 1 ABC transporters - Organism-specific
map00970 1 Aminoacyl-tRNA synthetases
hsa04080 1 Neuroactive ligand-receptor interaction
===[module15]===(27)===
map00790 12 Folate biosynthesis
map00740 9 Riboflavin metabolism
map00670 8 One carbon pool by folate
map00630 3 Glyoxylate and dicarboxylate metabolism
map00680 2 Methane metabolism
map00970 2 Aminoacyl-tRNA synthetases
map00230 2 Purine metabolism
map00260 2 Glycine, serine and threonine metabolism
map00623 1 2,4-Dichlorobenzoate degradation
vch05110 1 Cholera - Infection
hsa04540 1 Gap junction
===[module14]===(19)===
map00564 13 Glycerophospholipid metabolism
map00561 7 Glycerolipid metabolism
map00260 3 Glycine, serine and threonine metabolism
map04070 3 Phosphatidylinositol signaling system

map00051 2 Fructose and mannose metabolism
hsa00563 1 Glycosylphosphatidylinositol(GPI)-anchor
vch02011 1 ABC transporters - Organism-specific
map00562 1 Inositol phosphate metabolism
bha02011 1 ABC transporters - Organism-specific
map02010 1 ABC transporters - General
map00440 1 Aminophosphonate metabolism
map00052 1 Galactose metabolism
map00680 1 Methane metabolism
eco02011 1 ABC transporters - Organism-specific
sce00563 1 Glycosylphosphatidylinositol(GPI)-anchor
hsa04540 1 Gap junction
===[module19]===(33)===
map00620 11 Pyruvate metabolism
map00010 9 Glycolysis / Gluconeogenesis
map00710 7 Carbon fixation
map00720 6 Reductive carboxylate cycle (CO2 fixation)
map00020 6 Citrate cycle (TCA cycle)
map00040 5 Pentose and glucuronate interconversions
map00052 5 Galactose metabolism
map00630 5 Glyoxylate and dicarboxylate metabolism
map00030 5 Pentose phosphate pathway
map00430 5 Taurine and hypotaurine metabolism
map00252 4 Alanine and aspartate metabolism
eco02060 4 Phosphotransferase system (PTS)
map00051 4 Fructose and mannose metabolism
map00260 4 Glycine, serine and threonine metabolism
map00362 4 Benzoate degradation via hydroxylation
bsu02060 3 Phosphotransferase system (PTS)
map00031 3 Inositol metabolism
map00272 3 Cysteine metabolism
map00440 3 Aminophosphonate metabolism
map00100 3 Biosynthesis of steroids
map00400 3 Phenylalanine, tyrosine and tryptophan biosynthesis
mpn02060 3 Phosphotransferase system (PTS)
map00680 3 Methane metabolism
map00642 3 Ethylbenzene degradation
map00622 3 Toluene and xylene degradation
map00643 2 Styrene degradation
map00330 2 Arginine and proline metabolism
bbu02060 2 Phosphotransferase system (PTS)
map00251 2 Glutamate metabolism
map00650 2 Butanoate metabolism
map00750 2 Vitamin B6 metabolism
mge02060 2 Phosphotransferase system (PTS)
map00770 2 Pantothenate and CoA biosynthesis
map00920 2 Sulfur metabolism
map00290 2 Valine, leucine and isoleucine biosynthesis
hin02060 2 Phosphotransferase system (PTS)
map00625 2 Tetrachloroethene degradation
map00564 2 Glycerophospholipid metabolism
map00480 2 Glutathione metabolism
map00540 2 Lipopolysaccharide biosynthesis
map00360 2 Phenylalanine metabolism
map00640 2 Propanoate metabolism
map00660 2 C5-Branched dibasic acid metabolism
map00380 2 Tryptophan metabolism
map00629 2 Carbazole degradation
map00628 2 Fluorene degradation
map00627 2 1,4-Dichlorobenzene degradation
map00621 2 Biphenyl degradation
map00730 2 Thiamine metabolism
eco02011 1 ABC transporters - Organism-specific
map00271 1 Methionine metabolism
map00331 1 Clavulanic acid biosynthesis
map00253 1 Tetracycline biosynthesis
map00311 1 Penicillins and cephalosporins biosynthesis
map00532 1 Chondroitin / Heparan sulfate biosynthesis
map00350 1 Tyrosine metabolism
map00632 1 Benzoate degradation via CoA ligation
map00473 1 D-Alanine metabolism
map00062 1 Fatty acid elongation in mitochondria
map00410 1 beta-Alanine metabolism
map00071 1 Fatty acid metabolism
map00561 1 Glycerolipid metabolism
map00300 1 Lysine biosynthesis
map00450 1 Selenoamino acid metabolism
map00522 1 Biosynthesis of 12-, 14- and 16-membered macrolides
map00053 1 Ascorbate and aldarate metabolism
map00760 1 Nicotinate and nicotinamide metabolism
map00072 1 Synthesis and degradation of ketone bodies
hsa04930 1 Type II diabetes mellitus
map00641 1 3-Chloroacrylic acid degradation
map00280 1 Valine, leucine and isoleucine degradation
map00626 1 Nitrobenzene degradation
map00624 1 1- and 2-Methylnaphthalene degradation
map00310 1 Lysine degradation
map01056 1 Biosynthesis of type II polyketide backbone
map00970 1 Aminoacyl-tRNA synthetases
map00530 1 Aminosugars metabolism
map00061 1 Fatty acid biosynthesis
===[module18]===(7)===
map00640 4 Propanoate metabolism
map00280 4 Valine, leucine and isoleucine degradation
map00660 2 C5-Branched dibasic acid metabolism
map00642 1 Ethylbenzene degradation
map00522 1 Biosynthesis of 12-, 14- and 16-membered macrolides
map01056 1 Biosynthesis of type II polyketide backbone
map00410 1 beta-Alanine metabolism
===[module9]===(5)===
map00040 4 Pentose and glucuronate interconversions
map00630 2 Glyoxylate and dicarboxylate metabolism
map00625 1 Tetrachloroethene degradation
map00750 1 Vitamin B6 metabolism
map00790 1 Folate biosynthesis
===[module8]===(9)===
map00051 7 Fructose and mannose metabolism
map00620 3 Pyruvate metabolism
map00561 1 Glycerolipid metabolism
map00643 1 Styrene degradation
map00010 1 Glycolysis / Gluconeogenesis
map00640 1 Propanoate metabolism
===[module7]===(3)===
map00473 3 D-Alanine metabolism
map00550 2 Peptidoglycan biosynthesis
map00252 2 Alanine and aspartate metabolism
map00450 1 Selenoamino acid metabolism
map00720 1 Reductive carboxylate cycle (CO2 fixation)
ana02011 1 ABC transporters - Organism-specific
map00430 1 Taurine and hypotaurine metabolism
map00272 1 Cysteine metabolism
syn02011 1 ABC transporters - Organism-specific
map00710 1 Carbon fixation
map00970 1 Aminoacyl-tRNA synthetases
map00750 1 Vitamin B6 metabolism
===[module6]===(34)===
map00260 11 Glycine, serine and threonine metabolism
map00271 8 Methionine metabolism
map00480 8 Glutathione metabolism
map00450 6 Selenoamino acid metabolism
map00920 4 Sulfur metabolism
map00970 3 Aminoacyl-tRNA synthetases
map00330 2 Arginine and proline metabolism
map00251 2 Glutamate metabolism
map00680 2 Methane metabolism
map00640 2 Propanoate metabolism
map00310 1 Lysine degradation
map00230 1 Purine metabolism

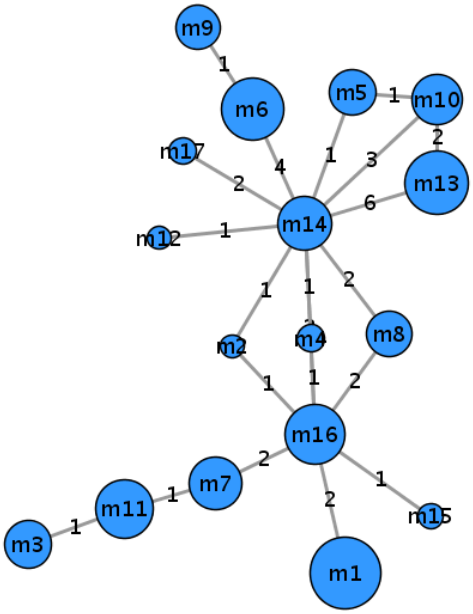
ana02011 1 ABC transporters - Organism-specific
map00272 1 Cysteine metabolism
map00120 1 Bile acid biosynthesis
map00290 1 Valine, leucine and isoleucine biosynthesis
map02010 1 ABC transporters - General
map00860 1 Porphyrin and chlorophyll metabolism
map00220 1 Urea cycle and metabolism of amino groups
map00300 1 Lysine biosynthesis
syn02011 1 ABC transporters - Organism-specific
map00660 1 C5-Branched dibasic acid metabolism
map00460 1 Cyanoamino acid metabolism
map00910 1 Nitrogen metabolism
map00620 1 Pyruvate metabolism
hsa04080 1 Neuroactive ligand-receptor interaction
===[module5]===(10)===
map00040 8 Pentose and glucuronate interconversions
map00500 3 Starch and sucrose metabolism
map00860 2 Porphyrin and chlorophyll metabolism
map00053 2 Ascorbate and aldarate metabolism
map00562 1 Inositol phosphate metabolism
map00520 1 Nucleotide sugars metabolism
map00030 1 Pentose phosphate pathway
===[module4]===(25)===
map00630 10 Glyoxylate and dicarboxylate metabolism
map00260 7 Glycine, serine and threonine metabolism
map00053 6 Ascorbate and aldarate metabolism
map00272 4 Cysteine metabolism
map00010 4 Glycolysis / Gluconeogenesis
map00020 3 Citrate cycle (TCA cycle)
map00720 3 Reductive carboxylate cycle (CO2 fixation)
map00230 2 Purine metabolism
map00920 2 Sulfur metabolism
map00710 2 Carbon fixation
map00561 2 Glycerolipid metabolism
map00660 2 C5-Branched dibasic acid metabolism
ana02011 1 ABC transporters - Organism-specific
map00271 1 Methionine metabolism
map00330 1 Arginine and proline metabolism
map00252 1 Alanine and aspartate metabolism
map00251 1 Glutamate metabolism
map00631 1 1,2-Dichloroethane degradation
map00450 1 Selenoamino acid metabolism
syn02011 1 ABC transporters - Organism-specific
map00680 1 Methane metabolism
map00361 1 gamma-Hexachlorocyclohexane degradation
map00460 1 Cyanoamino acid metabolism
map00627 1 1,4-Dichlorobenzene degradation
map00600 1 Glycosphingolipid metabolism
map00970 1 Aminoacyl-tRNA synthetases
===[module3]===(2)===
map00260 1 Glycine, serine and threonine metabolism
===[module2]===(21)===
map00400 19 Phenylalanine, tyrosine and tryptophan biosynthesis
map00360 3 Phenylalanine metabolism
map00970 3 Aminoacyl-tRNA synthetases
map00401 3 Novobiocin biosynthesis
map00380 3 Tryptophan metabolism
map00350 2 Tyrosine metabolism
map00362 2 Benzoate degradation via hydroxylation
map00950 2 Alkaloid biosynthesis I
map01055 2 Biosynthesis of vancomycin group antibiotics
map00790 1 Folate biosynthesis
map00632 1 Benzoate degradation via CoA ligation
ana02011 1 ABC transporters - Organism-specific
map00960 1 Alkaloid biosynthesis II
map00130 1 Ubiquinone biosynthesis
map00629 1 Carbazole degradation
map00623 1 2,4-Dichlorobenzoate degradation
syn02011 1 ABC transporters - Organism-specific
map01053 1 Biosynthesis of siderophore group nonribosomal
map00901 1 Indole and ipecac alkaloid biosynthesis
===[module1]===(34)===
map00300 13 Lysine biosynthesis
map00550 9 Peptidoglycan biosynthesis
map00530 7 Aminosugars metabolism
map00410 5 beta-Alanine metabolism
map00252 4 Alanine and aspartate metabolism
map00310 3 Lysine degradation
map00471 3 D-Glutamine and D-glutamate metabolism
map00260 3 Glycine, serine and threonine metabolism
map00970 3 Aminoacyl-tRNA synthetases
eco02011 2 ABC transporters - Organism-specific
map00960 2 Alkaloid biosynthesis II
map02010 2 ABC transporters - General
map00031 2 Inositol metabolism
map00640 2 Propanoate metabolism
map00760 2 Nicotinate and nicotinamide metabolism
map00460 2 Cyanoamino acid metabolism
map00910 2 Nitrogen metabolism
map00330 2 Arginine and proline metabolism
hsa04080 2 Neuroactive ligand-receptor interaction
map00253 1 Tetracycline biosynthesis
map02031 1 Bacterial chemotaxis - Organism-specific
map00770 1 Pantothenate and CoA biosynthesis
map00710 1 Carbon fixation
map00340 1 Histidine metabolism
map00220 1 Urea cycle and metabolism of amino groups
map00780 1 Biotin metabolism
map00540 1 Lipopolysaccharide biosynthesis
map00240 1 Pyrimidine metabolism
map00641 1 3-Chloroacrylic acid degradation
pae02011 1 ABC transporters - Organism-specific
sce00563 1 Glycosylphosphatidylinositol(GPI)-anchor
===[module26]===(4)===
map00632 3 Benzoate degradation via CoA ligation
map00650 3 Butanoate metabolism
map00310 2 Lysine degradation
map00280 2 Valine, leucine and isoleucine degradation
map00072 2 Synthesis and degradation of ketone bodies
map00071 2 Fatty acid metabolism
map00640 2 Propanoate metabolism
map00380 2 Tryptophan metabolism
map00350 1 Tyrosine metabolism
map00643 1 Styrene degradation
map00620 1 Pyruvate metabolism
map02020 1 Two-component system - General
===[module27]===(29)===
map00240 29 Pyrimidine metabolism
map00550 2 Peptidoglycan biosynthesis
hsa04080 2 Neuroactive ligand-receptor interaction
map00770 1 Pantothenate and CoA biosynthesis
map00252 1 Alanine and aspartate metabolism
map00410 1 beta-Alanine metabolism
===[module24]===(5)===
map00051 5 Fructose and mannose metabolism
map00510 1 N-Glycan biosynthesis
bsu02060 1 Phosphotransferase system (PTS)
eco02060 1 Phosphotransferase system (PTS)
===[module25]===(4)===
map00100 4 Biosynthesis of steroids
===[module22]===(25)===
map00220 17 Urea cycle and metabolism of amino groups
map00330 16 Arginine and proline metabolism
map02010 7 ABC transporters - General
eco02011 6 ABC transporters - Organism-specific
map00251 5 Glutamate metabolism
ana02011 4 ABC transporters - Organism-specific
map00471 4 D-Glutamine and D-glutamate metabolism
map00970 4 Aminoacyl-tRNA synthetases
map00410 3 beta-Alanine metabolism

pae02011 3 ABC transporters - Organism-specific
hin02011 3 ABC transporters - Organism-specific
map00910 3 Nitrogen metabolism
map00230 2 Purine metabolism
map00340 2 Histidine metabolism
syn02011 2 ABC transporters - Organism-specific
map00240 2 Pyrimidine metabolism
vch02011 2 ABC transporters - Organism-specific
map00472 2 D-Arginine and D-ornithine metabolism
map00331 1 Clavulanic acid biosynthesis
map00252 1 Alanine and aspartate metabolism
map00750 1 Vitamin B6 metabolism
hsa04540 1 Gap junction
map00650 1 Butanoate metabolism
map00960 1 Alkaloid biosynthesis II
map00860 1 Porphyrin and chlorophyll metabolism
bsu02011 1 ABC transporters - Organism-specific
map00660 1 C5-Branched dibasic acid metabolism
map00401 1 Novobiocin biosynthesis
map00480 1 Glutathione metabolism
hsa04080 1 Neuroactive ligand-receptor interaction
===[module23]===(4)===
map00330 4 Arginine and proline metabolism
===[module20]===(23)===
map00770 16 Pantothenate and CoA biosynthesis
map00290 9 Valine, leucine and isoleucine biosynthesis
map00280 4 Valine, leucine and isoleucine degradation
eco02011 2 ABC transporters - Organism-specific
ana02011 2 ABC transporters - Organism-specific
map02010 2 ABC transporters - General
map00970 2 Aminoacyl-tRNA synthetases
pae02011 2 ABC transporters - Organism-specific
map00071 2 Fatty acid metabolism
map00311 1 Penicillins and cephalosporins biosynthesis
map00230 1 Purine metabolism
map00650 1 Butanoate metabolism
map00640 1 Propanoate metabolism
map00620 1 Pyruvate metabolism
map00540 1 Lipopolysaccharide biosynthesis
syn02011 1 ABC transporters - Organism-specific
map00532 1 Chondroitin / Heparan sulfate biosynthesis
map00061 1 Fatty acid biosynthesis
map00020 1 Citrate cycle (TCA cycle)
map00410 1 beta-Alanine metabolism
map00920 1 Sulfur metabolism
===[module21]===(18)===
map00040 8 Pentose and glucuronate interconversions
map00030 7 Pentose phosphate pathway
map00710 5 Carbon fixation
map00230 3 Purine metabolism
eco02011 3 ABC transporters - Organism-specific
map00520 3 Nucleotide sugars metabolism
map02010 3 ABC transporters - General
bha02011 2 ABC transporters - Organism-specific
map00680 2 Methane metabolism
bsu02011 2 ABC transporters - Organism-specific
map00500 2 Starch and sucrose metabolism
map02031 1 Bacterial chemotaxis - Organism-specific
vch02011 1 ABC transporters - Organism-specific
hin02011 1 ABC transporters - Organism-specific
map00271 1 Methionine metabolism
map00053 1 Ascorbate and aldarate metabolism
vch05110 1 Cholera - Infection
pae02011 1 ABC transporters - Organism-specific
map00540 1 Lipopolysaccharide biosynthesis
map00630 1 Glyoxylate and dicarboxylate metabolism
===[module28]===(7)===
map00061 5 Fatty acid biosynthesis
map00640 1 Propanoate metabolism
map00522 1 Biosynthesis of 12-, 14- and 16-membered macrolides
map00620 1 Pyruvate metabolism
map00253 1 Tetracycline biosynthesis
map01056 1 Biosynthesis of type II polyketide backbone
map01055 1 Biosynthesis of vancomycin group antibiotics
map00410 1 beta-Alanine metabolism

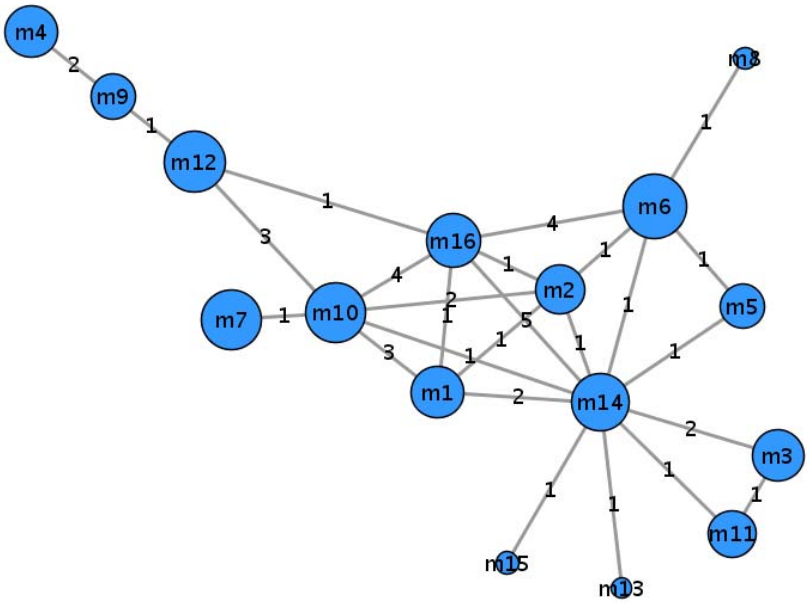
Modules



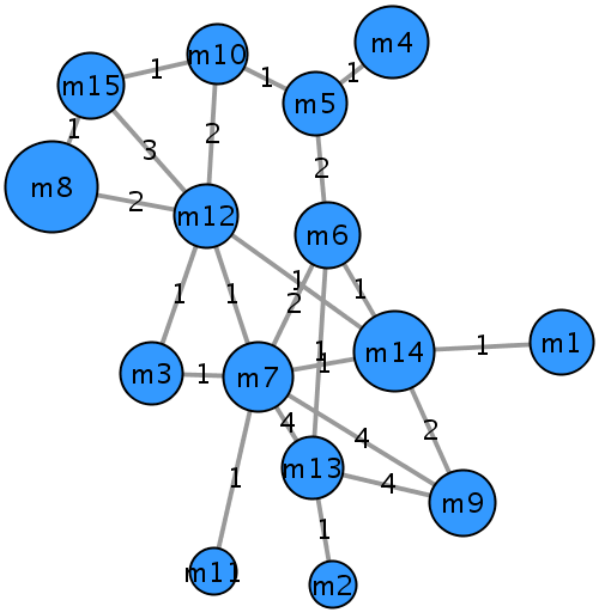
Module decomposition for *A. pernix*.



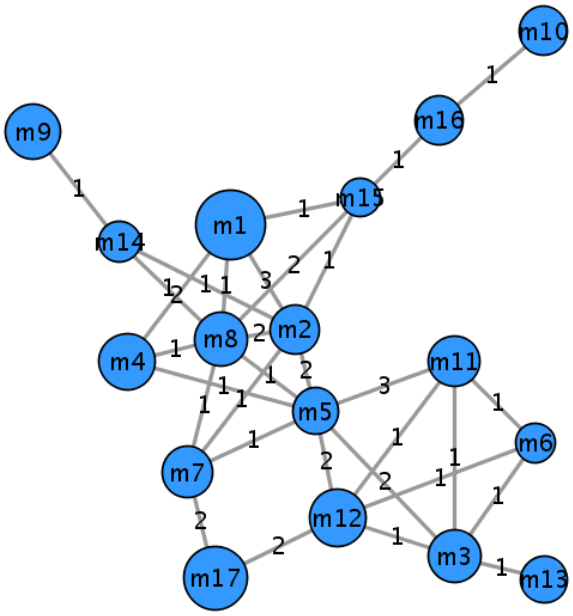
Module decomposition for *B. subtilis*.



Module decomposition for *E. coli*.



Module decomposition for *S. cerevisiae*.



Module decomposition for *H. sapiens*.

References

Biochemical Pathways. http://www.expasy.ch/cgi-bin/show_thumbnails.pl . 2006a.
Ref Type: Electronic Citation

Wikipedia. http://wikipedia.org . 2006b.
Ref Type: Electronic Citation

Albert R, Barabasi AL & Jeong H 2000 Power-law distribution of the World Wide Web. *Science* **287** 2115.

Amaral LA, Scala A, Barthelemy M & Stanley HE 10-10-2000 Classes of small-world networks. *Proc.Natl.Acad.Sci.U.S.A* **97** 11149-11152.

Bagrow JP & Bollt EM 2005 Local method for detecting communities. *Phys.Rev.E.Stat.Nonlin.Soft.Matter Phys.* **72** 046108.

Barabasi AL & Albert R 15-10-1999 Emergence of scaling in random networks. *Science* **286** 509-512.

Battail, C. Systems Biology. http://strc.herts.ac.uk/bio/christophe/Systems_biology.htm . 2005.
Ref Type: Electronic Citation

Brandes, U. Faster Evaluation of Shortest-Path Based Centrality Indices. [120]. 2000. Konstanzer Schriften in Mathematik und Informatik.
Ref Type: Serial (Book,Monograph)

Comellas, F. and Miralles, A. An efficient and fast local algorithm to identify clusters in networks. 2005.
Ref Type: Unpublished Work

Edwards JS, Covert M & Palsson B 2002 Metabolic modelling of microbes: the flux-balance approach. *Environ.Microbiol.* **4** 133-140.

Forst CV & Schulten K 2001 Phylogenetic analysis of metabolic pathways. *J.Mol.Evol.* **52** 471-489.

Gagneur J, Jackson DB & Casari G 22-5-2003 Hierarchical analysis of dependency in metabolic networks. *Bioinformatics.* **19** 1027-1034.

Girvan M & Newman ME 11-6-2002 Community structure in social and biological networks. *Proc.Natl.Acad.Sci.U.S.A* **99** 7821-7826.

Guimera R & Nunes Amaral LA 24-2-2005 Functional cartography of complex metabolic networks. *Nature* **433** 895-900.

Hartwell LH, Hopfield JJ, Leibler S & Murray AW 2-12-1999 From molecular to modular cell biology. *Nature* **402** C47-C52.

Holme P 2005 Core-periphery organization of complex networks. *Phys.Rev.E.Stat.Nonlin.Soft.Matter Phys.* **72** 046111.

Holme P, Huss M & Jeong H 1-3-2003 Subnetwork hierarchies of biochemical pathways. *Bioinformatics.* **19** 532-538.

Ikeda, K. Shortest Path Problem: Dijkstra's Algorithm. http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/Dijkstra.shtml . 7-6-2000.
Ref Type: Electronic Citation

Jeong H, Tombor B, Albert R, Oltvai ZN & Barabasi AL 5-10-2000 The large-scale organization of metabolic networks. *Nature* **407** 651-654.

Jungnickel D 2002 *Graphs, Networks and Algorthms*.

Junker BH & Schreiber F 2006 *Biological Network Analysis*. Willey.

Kanehisa M & Goto S 1-1-2000 KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **28** 27-30.

Kirkpatrick S, Gelatt CD & Vecchi MP 1983 Optimization by simulated annealing. *Science* 671-680.

Kitano H 2002b Looking beyond the details: a rise in system-oriented approaches in genetics and molecular biology. *Curr.Genet.* **41** 1-10.

Kitano H 14-11-2002a Computational systems biology. *Nature* **420** 206-210.

Krebs, Valdis. How to do Social Network Analysis. http://www.orgnet.com/sna.html . 2006.
Ref Type: Electronic Citation

Ma H & Zeng AP 22-1-2003 Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms. *Bioinformatics* **19** 270-277.

Ma HW & Zeng AP 2004 Phylogenetic comparison of metabolic capacities of organisms at genome level. *Mol.Phylogenet.Evol.* **31** 204-213.

Ma HW, Zhao XM, Yuan YJ & Zeng AP 12-8-2004 Decomposition of metabolic network into functional modules based on the global connectivity structure of reaction graph. *Bioinformatics* **20** 1870-1876.

Neidhardt FC, Ingraham JL & Schaechter M 1990 *Physiology of the bacterial cell: a molecular approach*. Sinauer Associates, Inc. Publishers.

Newman ME 16-1-2001 The structure of scientific collaboration networks. *Proc.Natl.Acad.Sci.U.S.A* **98** 404-409.

Newman ME 2004 Fast algorithm for detecting community structure in networks. *Phys.Rev.E.Stat.Nonlin.Soft.Matter Phys.* **69** 066133.

Newman ME & Girvan M 2004 Finding and evaluating community structure in networks. *Phys.Rev.E.Stat.Nonlin.Soft.Matter Phys.* **69** 026113.

Newman ME, Watts DJ & Strogatz SH 19-2-2002 Random graph models of social networks. *Proc.Natl.Acad.Sci.U.S.A* **99 Suppl 1** 2566-2572.

Palsson BO, Price ND & Papin JA 2003 Development of network-based pathway definitions: the need to analyze real metabolic networks. *Trends Biotechnol.* **21** 195-198.

Papin JA, Price ND & Palsson BO 2002 Extreme pathway lengths and reaction participation in genome-scale metabolic networks. *Genome Res.* **12** 1889-1900.

Podani J, Oltvai ZN, Jeong H, Tombor B, Barabasi AL & Szathmary E 2001 Comparable system-level organization of Archaea and Eukaryotes. *Nat.Genet.* **29** 54-56.

Price ND, Reed JL, Papin JA, Famili I & Palsson BO 2003 Analysis of metabolic capabilities using singular value decomposition of extreme pathway matrices. *Biophys.J.* **84** 794-804.

Ravasz E & Barabasi AL 2003 Hierarchical organization in complex networks. *Phys.Rev.E.Stat.Nonlin.Soft.Matter Phys.* **67** 026112.

Ravasz E, Somera AL, Mongru DA, Oltvai ZN & Barabasi AL 30-8-2002 Hierarchical organization of modularity in metabolic networks. *Science* **297** 1551-1555.

Schilling CH, Edwards JS & Palsson BO 1999 Toward metabolic phenomics: analysis of genomic data using flux balances. *Biotechnol.Prog.* **15** 288-295.

Schilling CH, Letscher D & Palsson BO 7-4-2000 Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *J.Theor.Biol.* **203** 229-248.

Schilling CH & Palsson BO 7-4-2000 Assessment of the metabolic capabilities of Haemophilus influenzae Rd through a genome-scale pathway analysis. *J.Theor.Biol.* **203** 249-283.

Schuster S, Dandekar T & Fell DA 1999 Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends Biotechnol.* **17** 53-60.

Schuster S, Fell DA & Dandekar T 2000 A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nat.Biotechnol.* **18** 326-332.

Schuster S, Pfeiffer T, Moldenhauer F, Koch I & Dandekar T 2002 Exploring the pathway structure of metabolism: decomposition into subnetworks and application to Mycoplasma pneumoniae. *Bioinformatics.* **18** 351-361.

Scott J 2000 *Social Network Analysis: A Handbook*. London: Sage Publications.

Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B & Ideker T 2003 Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13** 2498-2504.

Stelling J, Klamt S, Bettenbrock K, Schuster S & Gilles ED 14-11-2002 Metabolic network structure determines key aspects of functionality and regulation. *Nature* **420** 190-193.

Strogatz SH 8-3-2001 Exploring complex networks. *Nature* **410** 268-276.

Wasserman S & Faust K 1994 *Social Network Analysis*. Cambridge: Cambridge University Press.

Watts DJ & Strogatz SH 4-6-1998 Collective dynamics of 'small-world' networks. *Nature* **393** 440-442.

Xiu ZL, Zeng AP & Deckwer WD 5-2-1998 Multiplicity and stability analysis of microorganisms in continuous culture: effects of metabolic overflow and growth inhibition. *Biotechnol.Bioeng.* **57** 251-261.

Lebenslauf

Márcio Rosa da Silva, M.Sc.

Siedlerstr. 20
38124 Braunschweig

Tel: 05 31 / 6 12 89 33

geboren am 21. Juni 1972 in Porto Alegre, Brasilien

Verheiratet

Akademische Berufslaufbahn

Seit März 1998	Dozent an der Universidade do Vale do Rio dos Sinos (UNISINOS) in São Leopoldo, Brasilien. (Seit August 2001 frei gestellt für die Promotion an der TU-Braunschweig).

Hochschulstudium

März 1996- Februar 1998	Universidade Federal de Santa Catarina Florianópolis, Brasilien Master in Elektroingenieurwissenschaft (Schwerpunkt: Biomedizinische Ingenieurwissenschaft)
März 1990- Januar 1996	Universidade Federal do Rio Grande do Sul Porto Alegre, Brasilien Studium in Elektroingenieurwissenschaft

Schulausbildung

März 1987- Dezember 1989	Colégio Anchieta Porto Alegre, Brasilien Sekundarschule
März 1979- Dezember 1986	Colégio Santa Inês Porto Alegre, Brasilien Grundschule